MetaLink Corporation

iceMASTER™ User's Manual

Document Version: 1.0 **Software Version: 3.0**

Table of Contents

Chapter 7: Command Reference 7-1

Chapter 8: Run-Time Considerations 8-1

Chapter 9: Troubleshooting 9-1

Appendix A: Tutorial A-1

Chapter 1: Preliminary Information

If you need to contact us:

Address:	**MetaLink Corporation**
	P.O. BOX 1329
	Chandler, AZ 85244-1329
	U.S.A.

For Customer or Technical Assistance:

Phone:	**(800) METAICE**
	(800) 638-2423
	(602) 926-0797
Fax:	**(602) 926-1198**
Telex:	**4998050 MTLNK**

Warranty Information

MetaLink Corporation makes no warranties other than those contained herein and MetaLink Corporation expressly disclaims any and all warranties of fitness for a particular purpose.

Disclaimer of All Warranties and Liability

MetaLink Corporation makes no warranties, either expressed or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability or fitness for any particular purpose. MetaLink Corporation software is sold or licensed "as is". In no event shall MetaLink Corporation be liable for incidental or consequential damages resulting from any defect in the software.

Limited Warranty

The following limited warranties shall apply:

Emulators

All electronics of MetaLink emulators are guaranteed against defects due to materials or workmanship for a ninety (90) day period from the invoice date. For registered iceMASTER units this period is extended to one (1) year from the invoice date. MetaLink emulators contain no user-serviceable components. **This warranty is voided and any unpaid balance is due immediately if an emulator chassis has been damaged or opened for any reason.**

Emulator Probe Assemblies

The iceMASTER emulator's probe assembly is guaranteed against defects due to materials or workmanship for a ninety (90) day period from the invoice date. For registered iceMASTER units this period is extended to one (1) year from the invoice date. This warranty does not include damage to the pins of the probe assembly. Some MetaLink probe assemblies are shipped with a special "Bondout" version of a device. MetaLink's warranty of this special device is limited to that of the manufacturer.

Unauthorized Service

No warranty extended by MetaLink shall apply to any goods which have been modified or altered by persons other than MetaLink's authorized personnel; to goods that are defective due to misuse, neglect, improper installation, soldering or accident; or to goods sold as "used".

Payment Requirement

The foregoing limited warranties shall not apply unless Buyer has paid in full for the MetaLink products. Updates to the emulator User's Manual and emulator Host Software are available free to Buyer upon request for a ninety (90) day period from the invoice date. For registered iceMASTER units, this period is extended to a one-year-period from the invoice date.

Controlling Law

MetaLink Corporation does business and is located solely in the state of Arizona. All orders or agreements and the rights of the parties hereunder shall be governed by the laws of the state of Arizona.

Liability

MetaLink's liability hereunder is expressly limited, at MetaLink's option, to either:

1) the repair or replacement of the iceMASTER emulator that meets **MetaLink's** Limited Warranty, or
2) to a credit to Buyer in an amount equal to the Purchase Price of the iceMASTER emulator.

In no event shall MetaLink Corporation be liable for any incidental or consequential damages, losses or expenses directly or indirectly arising from the sale, handling, installation or use of MetaLink Corporation products or from any other cause related thereto.

Restocking Charge

A restocking charge of 15% of the purchase price will be charged on all units returned within the ninety (90) day warranty period. Units being returned beyond the ninety (90) day warranty will have a negotiated restocking charge.

Extended Warranty

MetaLink offers an extended warranty. The above limited warranty terms may be extended beyond the ninety (90) day period for non-registered users or the one (1) year period for registered users, under our extended warranty program. You may purchase extensions in one-year-increments for a nominal fee, provided your unit is currently under warranty. If your warranty has expired, there will be a service fee to check out your unit and repair it, if necessary, before accepting the unit for extended coverage. Please contact MetaLink for more information and pricing.

Owner Registration

As you unpack your iceMASTER emulator take a moment to fill out and return the postage prepaid owner registration card. This card is located with the Host Software and Probe Card Software diskettes in the vinyl jacket at the front of this manual. It is very important that you return a completed card to us. This entitles you to extend the normal ninety (90) day warranty to one (1) year. Information from owner registration cards is used to distribute software revisions during the warranty period. Registered owners will be notified of any hardware upgrades, new products or product enhancements as they become available. We cannot be responsible for lost or misdirected registration cards, so please call us if you cannot find the card in your manual. We also suggest that you contact us about two weeks after returning the registration card to be sure we have received it.

Upgrade Policy

Any previously purchased MetaLink emulator may be upgraded to the iceMASTER series of emulators. Contact MetaLink (page 1-1) for details.

Warning

The MetaLink emulator has been shipped with the probe assembly inserted in a precision DIP, PLCC, or PGA socket with hardened pins, or with some similar method of protecting the pins of the probe assembly. We suggest that you protect the probe pin assembly at all times, even when in use. (When inserted in a target system the target socket should be capable of protecting the probe assembly pins). Failure to protect the pins of the probe assembly could result in a damaged and unusable probe assembly.

THE PROBE PINS ARE NOT COVERED UNDER THE METALINK CORPORATION EMULATOR WARRANTY.

Chapter 2: Introduction

About This Manual

The information presented in this manual applies only to the Host Software version 3.0 revision 12 and the iceMASTER series of products. Statements found in this manual should not be applied to earlier software or hardware products other than those specified in the warranty. If you have an earlier version of either the Host Software, manual or hardware, do not mix versions. Each version has many unique features and may not be compatible with similar products.

The manual is organized into several sections which include:

Preliminary Information contains important copyright, trademark, and warranty information, handling information, owner registration and upgrade information and the table of contents.

Introduction which you are now reading, contains a description of each section of the manual as well as other information to get you started.

Hardware Installation contains a description of the hardware and how to install it.

Probe Card Reference contains detailed information about each probe card including jumper configuration and available modes of operation.

Software Installation contains information and requirements for installing the software on your hard disk.

Software Guide contains a description of the terminology used in the manual, explains the layout of the screen and a description of the features of the Host Software.

Command Reference contains information on how to use each command in the Host Software.

Run-Time Considerations contains a description of some things to be aware of while working in an emulation environment.

Troubleshooting contains a description of what to look for and how to go about it should problems arise.

Appendices contain various topics related to the emulator, the Host Software, the Host Computer and the Tutorial.

This manual will show you how to use the iceMASTER emulator with your PC. This combination of iceMASTER and the PC is a powerful engineering development system.

The Emulator

The iceMASTER emulator is an in-circuit emulator controlled by an IBM PC (or compatible) running the PC-DOS/MS-DOS operating system. The iceMASTER emulator is an integral part of the development engineer's toolbox, with application in software development, hardware integration, manufacturing test and field service.

The emulator can be operated in a target system in place of the microcontroller, or independently in stand alone mode. Stand alone mode allows you to emulate hardware and/or execute code without a target system (provided no interaction with external devices is needed).

Hardware designers may use the emulator to develop and debug their designs. All available features of a given device are accessible interactively, as well as through user programs. Software designers have complete emulation capability as well. The emulator will execute your code just like the real part because it uses the real part for emulation.

Recommended References

Several additional references can be of help to you as you progress through the development process. The DOS reference guide for the version of DOS you are using provides helpful information on filename conventions and batch files. Batch files provide a simple method for invoking the Host Software with a minimum of effort. The data book and programmer's guide for the microcontroller you are using provide essential information. You will also need the programmer's manual for the development language you are using.

What You Need To Know

Throughout this manual it is presumed that you have a working knowledge of:

1) the family of microcontrollers you are emulating
2) the IBM PC (or compatible) as an engineering tool
3) a development language (Assembly Language, PL/M, or C)
4) IBM PC-DOS or Microsoft MS-DOS

A few of these topics are discussed in this manual as a means of illustrating a particular feature or facet of the iceMASTER emulator's abilities; however, basic programming knowledge and familiarity with the microcontroller architecture are assumed.

Getting Started

If you have not already done so, you should now read the warranty information (Chapter 1). From here you should perform the Hardware Installation (Chapter 3) and Software Installation (Chapter 5). Once the installation process is complete, we recommend you then read the Software Guide (Chapter 6) and then work through the Tutorial (Appendix A).

MetaLink iceMASTER emulators are compact, modular assemblies. Each subassembly is detachable by use of a mating connector. The subassemblies are:

1) iceMASTER emulator base chassis
2) Probe card
3) Probe card cable
4) Probe clip assembly
5) Power supply with cable (you may elect to furnish your own power supply)

Installation

Connect the 50 pin flat cable to both the emulator and the probe card. You should never turn the power to the emulator base on with the probe card disconnected!

Connect the RS-232 cable to the emulator and to the Host Computer. Be sure you are using Comm Port 1 or 2.

Connect the power supply to the emulator by inserting the power supply's connector into the emulator power receptacle. For safety, we recommend that all items in your system, including emulator, Host Computer and target, be connected to the same outlet. Different outlets though near one another may be connected to different circuits resulting in large potential differences between grounds.

Connect the probe clip assembly. (Optional).

At this time you should refer to the Probe Card Reference (Chapter 4) to locate the probe card you are installing. There are only a few basic functions requiring selection. Most of these are common to all of the MetaLink probe cards, though a few are probe card specific. Each probe card type has an illustration showing where the jumpers are located for that probe card and the page opposite the probe card illustration explains what each jumper selection does.

If you are going to use the probe card in a stand alone configuration with no target system attached, you need to set the XTAL jumpers to PC so that you are using the on-board oscillator. If you intend to follow the Tutorial (Appendix A), you should set all the jumpers to their default setting. After completing the Tutorial, you may change the settings to fit your application.

Before you proceed, please take a moment to familiarize yourself with the hardware and its controls. We especially recommend that you thoroughly understand the function of each jumper selection for future reference.

Host PC			Cable	Emulator Base Cable Connector	
	Cable Connector Pin		(Function)	(Male DB-25)	
Signal	PC / XT (Female DB-25)	PC / AT (Female DB-9)	(Direction)	Pin	Signal
TxD	2	3	(Data to ICE) ——>——>	2	RxD
RxD	3	2	(Data to Host) <——<——	3	TxD
RTS	4	7	(Reset ICE - active low) ——>——>	4	RTS
CTS	5	8	(ALE to Host) <——<——	5	CTS
DSR	6	6	(DSR to Host) <——<——	6	DSR
Ground	7	5	(DC Ground)	7	Ground
DTR	20	4	(Handshake) ——>——>	20	DTR

Figure 3-1. RS-232 Interface

The communication link to the Host Computer is based on the serial RS-232C specification. The serial baud rates are established entirely under Host Software control, so no adjustment is necessary to your serial ports baud rate. We neither use these pre-sets nor change them in any way.

The cable mates via a 25-pin male DB-25 connector on the cable at the emulator end. At the host end, the mating connector on the cable may be a 25-pin female DB-25 connector for the PC/XT type Host Computer or a 9-pin female DB-9 connector for the AT type Host Computer. A 25-pin cable is provided with the emulator. Adaptors are available to connect to 9-pin D connectors. Note that Pins 2 and 3 are reversed from their normal 25-pin D connector assignments in the 9-pin RS-232C interface of the PC AT.

Probe Card Cable

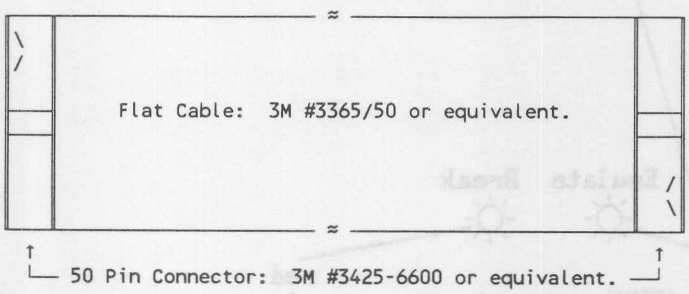


Figure 3-2. Probe Card Cable

On the probe card cable, IDC connectors mate with 50-pin ejector headers at either end for quick, easy assembly and disassembly of the emulator's primary parts. The ribbon cable can be reversed end-for-end with no difficulty, however, we recommend that you leave the red stripe denoting pin one to the right of the probe card's connector and to the left of the emulator base's connector, to avoid confusion later.

Probe Card Protection

It is advisable to use a high-quality DIP, PGA or PLCC socket (as appropriate) at all times to protect the hardened machined pins of the probe card. The machined pins of the probe card are not easily replaced nor are they covered by the warranty.

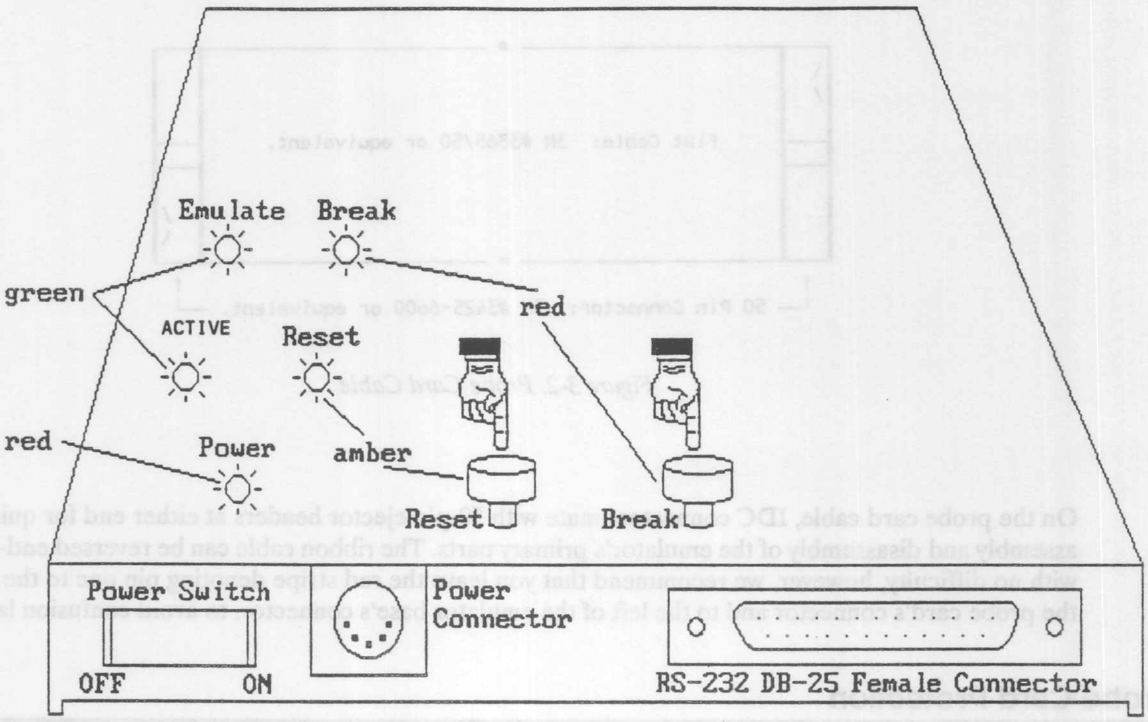


Figure 3-3. Emulator Chassis

Switches

Three switches are provided on the emulator chassis, as follows:

1) The rocker switch on the left side-panel is for **Power**.

2) The push-button labeled **Reset**, on top of the emulator, allows you to reset the probe card. Unless there is an independent reset-out signal provided by the microcontroller, there is no reset provided by this switch for your target system. If you require such a target system reset, you must provide it. Reset inputs to the microcontroller are valid.

3) Activating the push-button labeled **Break**, puts the emulator into Break condition. That is, it halts execution of your code and places you in interactive mode in the Host Software.

1) The red LED, labeled **Power**, is lit when power is applied to the emulator.

2) The green LED, labeled **Active**, is lit to indicate that ALE is strobing and will normally be ON unless the power-saving CMOS modes of operation are invoked.

3) The amber LED, labeled **Reset**, is lit when Reset is active. The Reset and Active LEDs are mutually exclusive. Only one should be active (ON) at a time, but if one of the 8051 family special modes is active, such as Idle or Power-down, it is normal for both LEDs to be OFF.

4) The red LED, labeled **Break**, will be ON when the emulator is in Break condition (interactive mode).

5) The green LED, labeled **Emulate**, will be ON when the emulator is in Emulation condition (user code is running).

Probe Clip Assembly

Probe Clip Assembly					
Schematic of Male Header on Probe Card	Orientation of Header and Silkscreened Labels as They Actually Appear on a Probe Card	Sche-matic Marking	Pin Function	Label Printed on Probe Card	Color of Corresponding Probe Clip
G G G 6 4 2 0 B T G NC 5 3 1	0 1 B T CLIPS	0 - 6	Probe Clips	CLIPS	All same color (but not: Green, Yellow or Black)
1 3 5 NC G T B 0 2 4 6 G G G	CLIPS T B 1 0	B	Break Input	B	Green
		T	Trigger Output	T	Yellow
		G	Ground	---	Black
		NC	Not Connected	---	---

14-Position Socket Connector: 3M 3385-6000 or equivalent

14-Conductor Flat Cable: Belden 9R28014 or equivalent (maximum length=12 in./30 cm.)

Test Clip, 13 Pieces: Pomona Test Type 4743 or equivalent

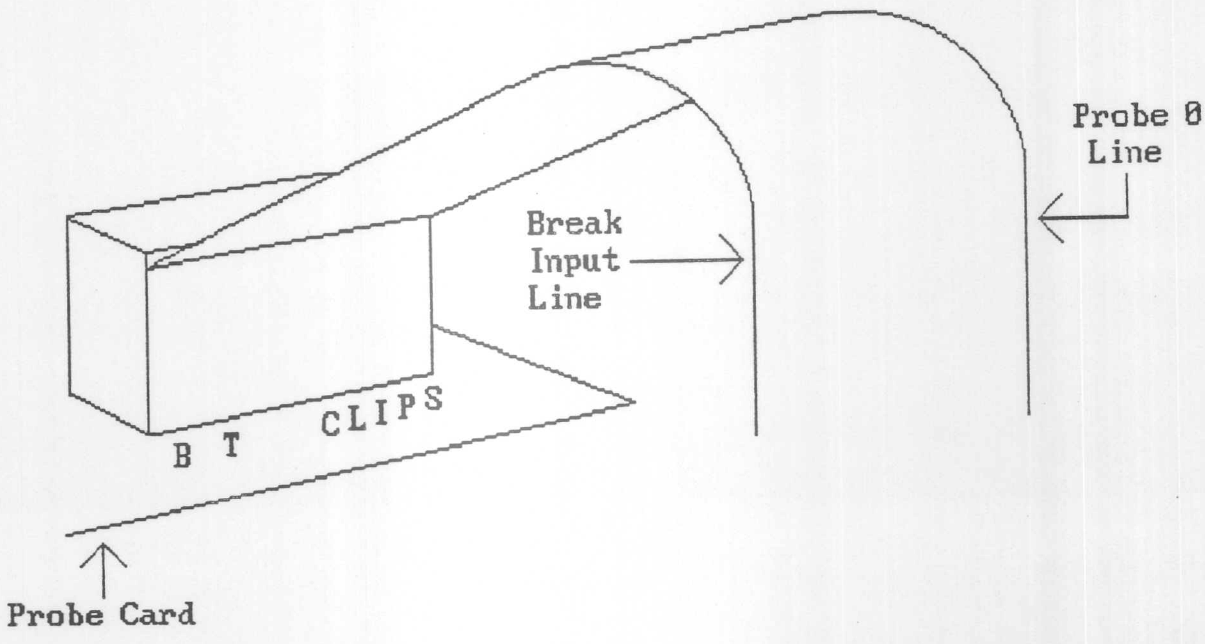


Figure 3-4. Probe Clip Assembly

The Probe Clip Assembly can be installed at your option. It is composed of seven input signal lines, four ground lines, one Trigger Output line, and one Break Input line.

The 14-pin female connector plugs in with the keying mark on the connector body-oriented to the silk-screened B at the corner of the male header on the Probe Card.

Input Signal Lines (CLIPS) There are seven user-selectable probe clip input signal lines which are inputs to a 74LS245 device on the probe card. These seven probe clip input signal lines can be used to select any TTL-level signal and record its state, once each cycle at a time corresponding to the falling edge of #PSEN (or where that signal would occur if it were present when operating in a ROM mode. These states are recorded in the trace buffer (see the *Break/Trace | View Trace* command on page 7-65 for Model 400 Emulators only).

Probe clip 0 is oriented as shown.

Break Input Signal (B) The Break Input signal is interfaced via a 74ACT245 device on the probe card. The Break Input is pulled up with a 2K ohm pull-up resistor. When this input is pulled low, the emulation cycle will break. The Break Input must remain active (LOW) for at least one instruction cycle.

Trigger Output Signal (T) The Trigger Output signal is interfaced via a 74ACT245 device on the probe card. The Trigger Output signal is normally HIGH and will strobe LOW (active) every time the Program Counter passes a Trace ON point (see the *Break/Trace | Set | Add | TRON* command on page 7-58 for Model 400 Emulators only). The Trigger Output will remain active for a minimum of one instruction cycle (until the next Opcode Fetch). Thus, for an instruction which uses two or more instruction cycles to complete, the Trace ON signal will remain active for that time.

Power Connector

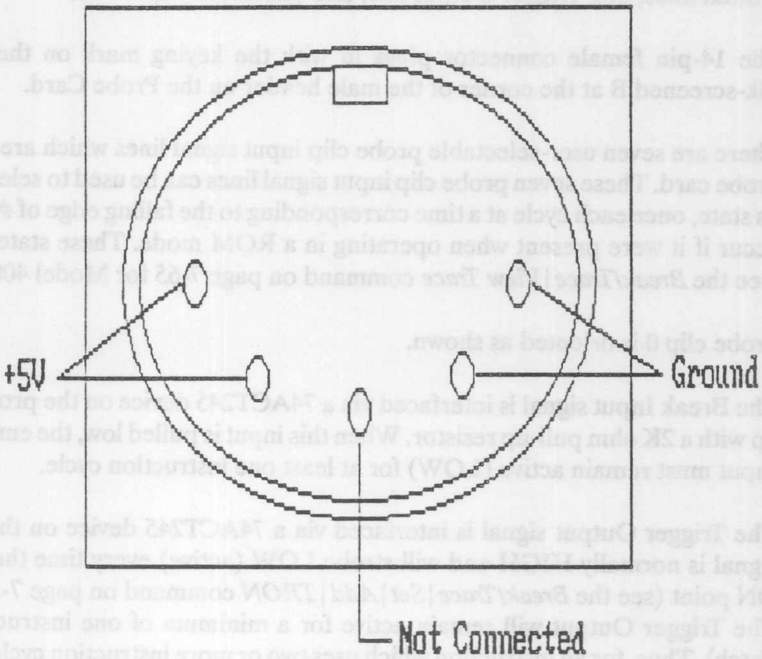


Figure 3-5. Power Connector (on emulator)

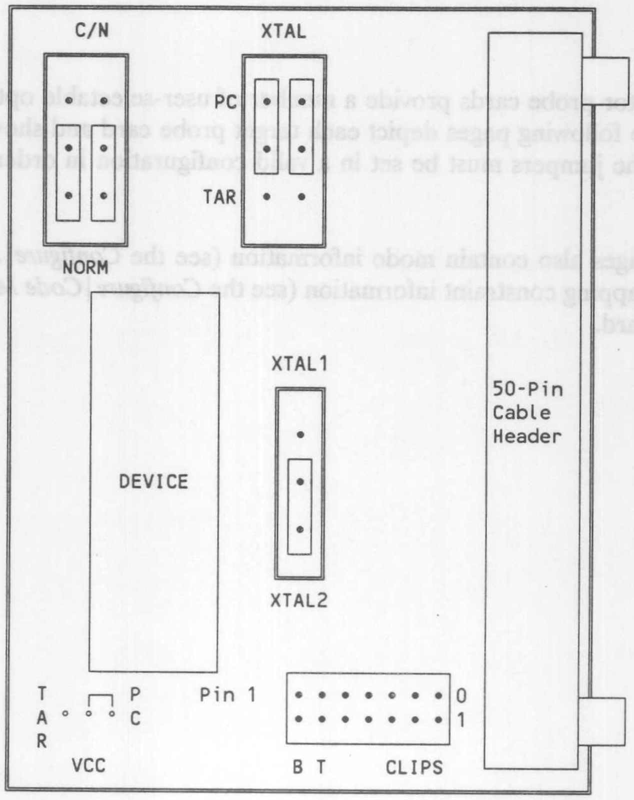
Power is supplied with a standard DIN audio connector similar to the keyboard connector found in many PC's. The power supply must provide +5 volts +/- 5%, at 1.5 amperes. The ripple voltage must be no greater than 50 millivolts, peak-to-peak.

Chapter 4: Probe Card Reference

MetaLink emulator probe cards provide a number of user-selectable options via jumper blocks on the probe cards. The following pages depict each target probe card and show the relative location of each jumper block. The jumpers must be set in a valid configuration in order for the emulator to function correctly.

The following pages also contain mode information (see the *Configure|Emulator|Mode* command on page 7-2) and mapping constraint information (see the *Configure|Code Memory* command on page 7-4) for each probe card.

8031, 8032, 8344, 80C51FA, 80C154, 80C321, 80C652, 80C851 and 80CL410 Probe Cards



8031,8032,8344,80C51FA,80C154,80C321,80CL410,80C652,80C851

XTAL (Oscillator) Selection Jumper Block

PC:	Probe card's crystal is used.
TAR:	Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

C/N - NORM Jumper Block

These two jumpers must always be set to the same relative position, either to C/N or NORM. The only time that the two C/N - NORM jumpers are in the C/N position is when the emulator is applied to a target built for an NMOS device and the user is using an external clock driver. At all other times, this jumper should be in the NORM position.

XTAL1 - XTAL2 Jumper Block

The XTAL1-XTAL2 jumper is set in the XTAL1 position only if an NMOS probe card is used in a target built for a CMOS controller that uses an external clock driver. At all other times, this jumper must be in the XTAL2 position. See the Map of Clock Jumpers at the end of this chapter (page 4-42).

Note that early versions of these probe cards were labeled DRIVER - XTAL. For these probe cards, DRIVER corresponds to XTAL1.

VCC (Device VCC) Jumper

The VCC jumper on this device is set at the factory to supply a VCC of +5V from the probe card. This jumper is not user selectable and must not be changed.

Device Orientation

The probe card is supplied with a probe head that has a DIP footprint connection for the target system. Pin one orientation of the DIP socket can be determined by viewing the probe head from the cable header (component) side. Pin one is in the lower right corner near the printed Pin one on the circuit board.

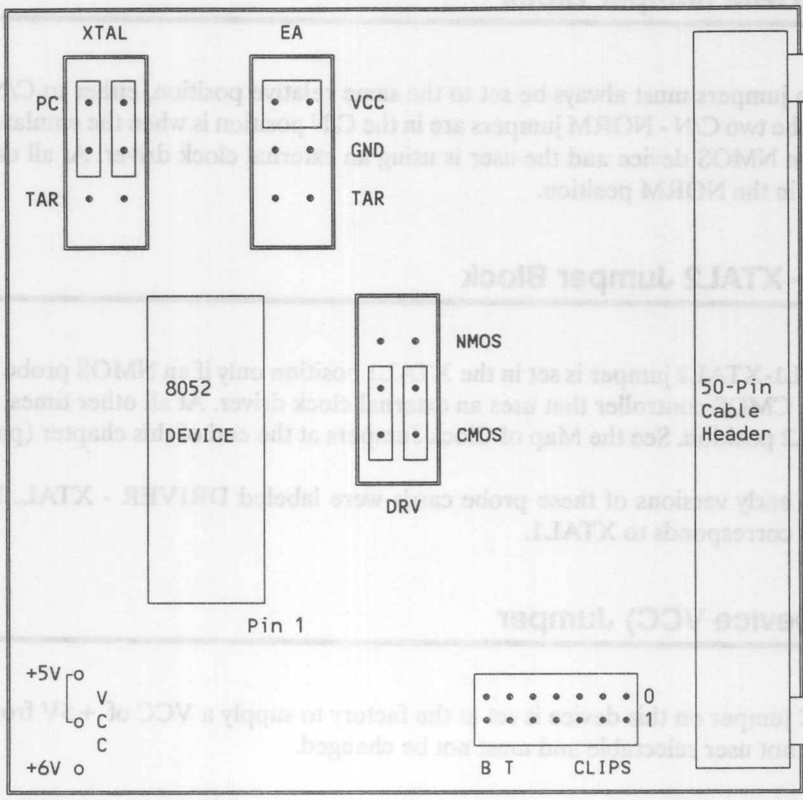
80C321 Modes Of Operation

Mode 1: Watchdog Timer Off

This is the default mode. In this mode of operation the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 2: Watchdog Timer On

In this mode of operation the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.



8052 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC:	Probe card's crystal is used.
TAR:	Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

EA (External Address) Selection Jumper Block

VCC:	Emulator controls EA pin (EA = HIGH = +5V) as ROM device.
GND:	Emulator controls EA pin (EA = LOW = 0V) as ROMless device.
TAR:	Target system controls EA pin.

Note that if you intend to use more than the amount of memory available in internal ROM, that is, if you intend to use this probe card in Rollover, you must configure the EA jumper to GND, and operate in ROMless mode (as you will be doing after Rollover).

NMOS - CMOS Jumper Block

The 8052 probe card is based on a CMOS controller. The only situation that would be "out of the ordinary" would be the use of an NMOS controller circuit with an external clock driver. In this case, the CMOS - NMOS jumpers should be set to the NMOS position. At all other times the jumpers should be set to the CMOS position. See the Map of Clock Jumpers at the end of this chapter (page 4-43).

VCC (Device VCC) Jumper

The VCC jumper on this device is set at the factory to supply a VCC of +5V from the probe card. This jumper is not user selectable and must not be changed.

Device Orientation

The 8052 probe card is supplied with a probe head that has a DIP footprint connection for the target system. Pin one orientation of the DIP socket can be determined by viewing the probe head from the cable header (component) side. Pin one is in the lower right corner near the printed Pin one on the circuit board.

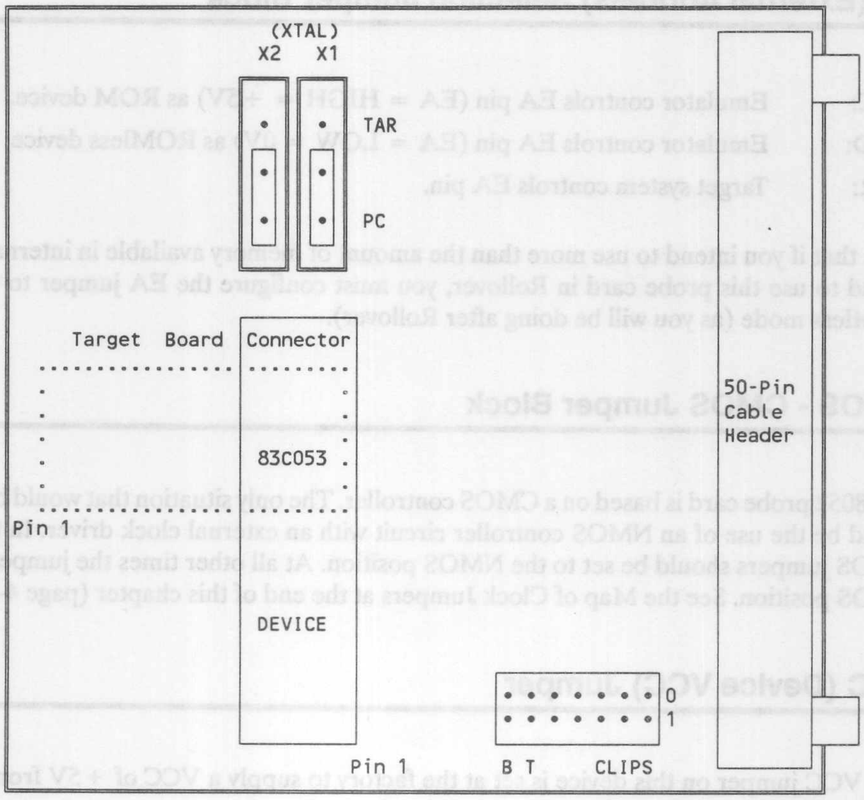
8052 Modes Of Operation

Mode 1: ROMless (803X) Operation, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

Mode 2: ROM (805X), /EA = High

This is the default mode. In this mode of operation, the emulator is configured to operate as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 16K) are internal and there is no access to code memory outside of the ROM address space so there can be no external code memory fetches. The ROM address space in code memory must be mapped to the emulator in this mode.



83C053 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.

TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

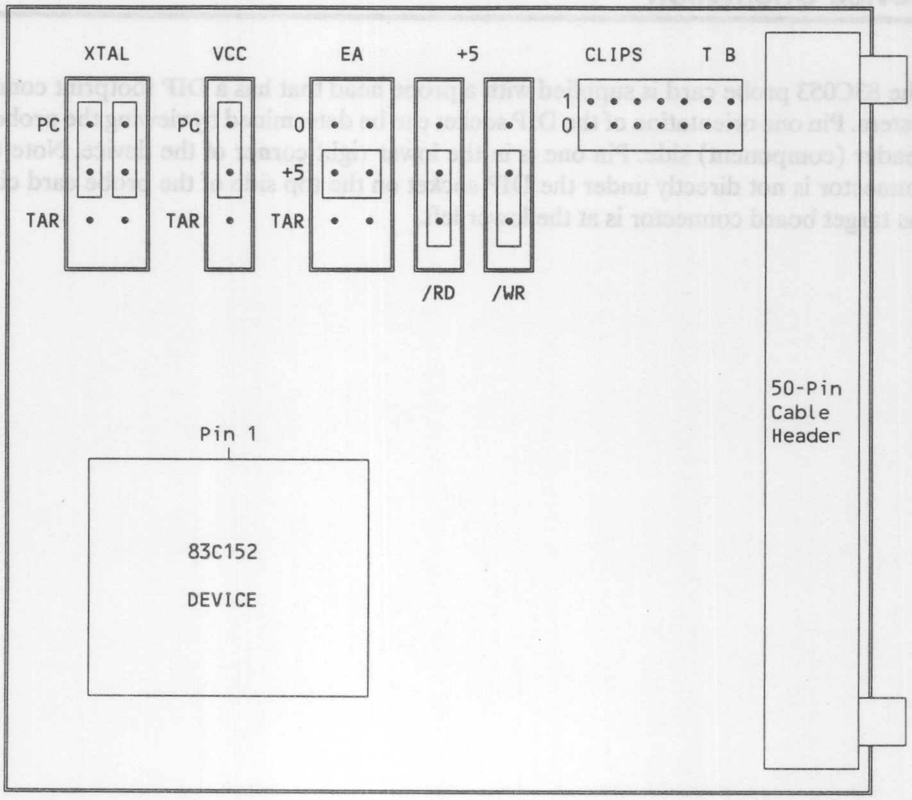
The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

Device Orientation

The 83C053 probe card is supplied with a probe head that has a DIP footprint connection for the target system. Pin one orientation of the DIP socket can be determined by viewing the probe head from the cable header (component) side. Pin one is in the lower right corner of the device. Note that the target board connector is not directly under the DIP socket on the top side of the probe card circuit board. Pin 1 on the target board connector is at the lower left.

83C152 Probe Card



83C152 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.

TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

VCC (Device VCC) Selection Jumper Block

PC: Probe card supplies a VCC of +5V.
TAR: Target system supplies VCC.

EA (External Address) Selection Jumper Block

+5: Emulator controls EA pin (EA = HIGH = +5V) as ROM device.
0: Emulator controls EA pin (EA = LOW = 0V) as ROMless device.
TAR: Target system controls EA pin.

/RD (Read Signal) Control Jumper Block

+5: Port 3.7 is an Input/Output pin.
/RD: Port 3.7 is used as Read Signal pin.

/WR (Write Signal) Control Jumper Block

+5: Port 3.6 is an Input/Output pin.
/WR: Port 3.6 is used as Write Signal pin.

Device Orientation

The 83C152 Probe card is supplied with a probe head that has a PLCC footprint connection for the target system. Pin one orientation of the PLCC socket can be determined by viewing the probe head from the cable header connector side. Pin one is on the upper side near the XTAL and VCC jumper blocks on the board.

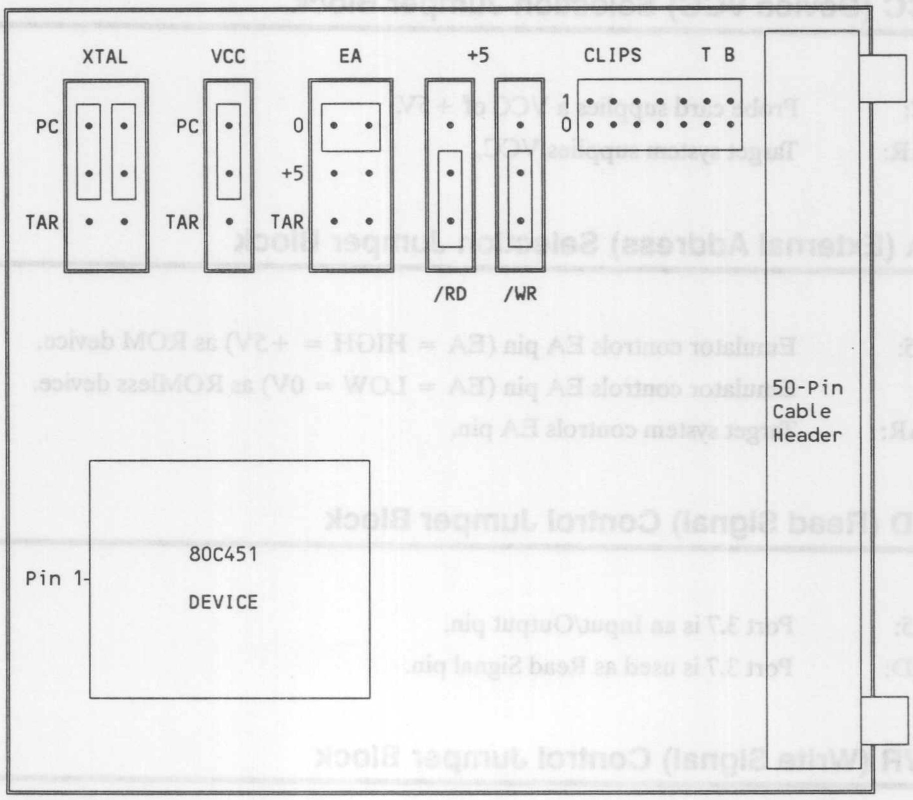
83C152 Modes Of Operation

Mode 1: ROMless Operation, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

Mode 2: ROM Operation, /EA = High

This is the default mode. In this mode of operation, the emulator is configured as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 8K) are internal and code memory fetches outside the ROM boundary are external. The ROM address space in code memory must be mapped to the emulator in this mode.



80C451 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.
TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

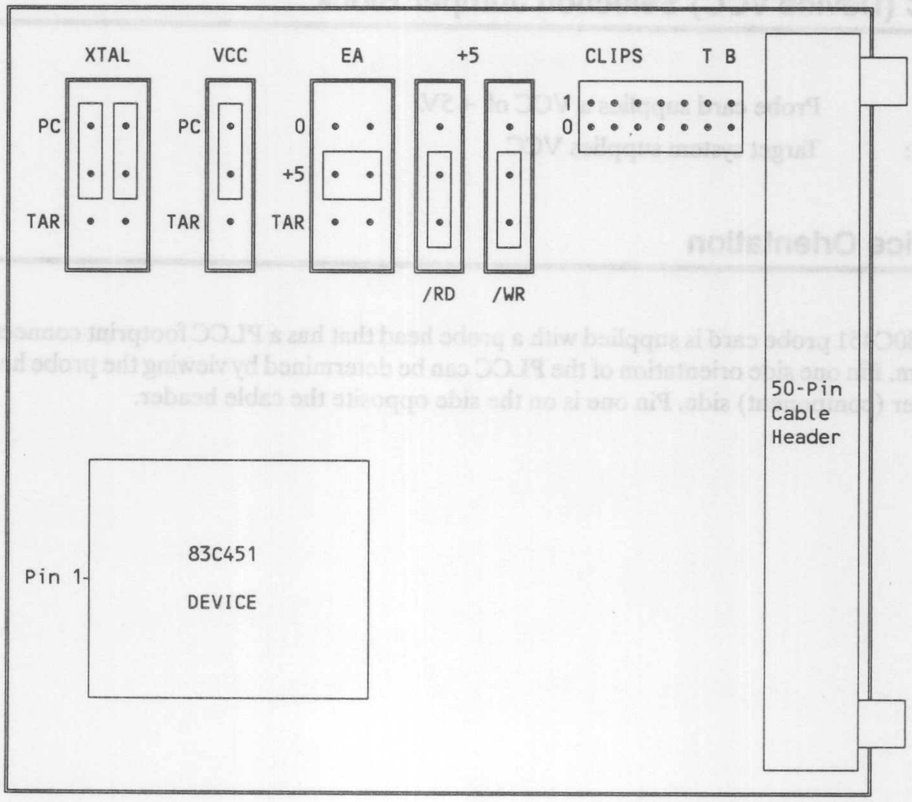
VCC (Device VCC) Selection Jumper Block

PC: Probe card supplies a VCC of +5V.

TAR: Target system supplies VCC.

Device Orientation

The 80C451 probe card is supplied with a probe head that has a PLCC footprint connection for the target system. Pin one side orientation of the PLCC can be determined by viewing the probe head from the cable header (component) side. Pin one is on the side opposite the cable header.



83C451 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC:	Probe card's crystal is used.
TAR:	Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

VCC (Device VCC) Selection Jumper Block

PC:	Probe card supplies a VCC of +5V.
TAR:	Target system supplies VCC.

EA (External Address) Selection Jumper Block

+5:	Emulator controls EA pin (EA = HIGH = +5V) as ROM device.
0:	Emulator controls EA pin (EA = LOW = 0V) as ROMless device
TAR:	Target system controls EA pin.

Device Orientation

The 83C451 probe card is supplied with a probe head that has a PLCC footprint connection for the target system. Pin one orientation of the PLCC socket can be determined by viewing the probe head from the cable header connector side. Pin one is on the side opposite the cable header.

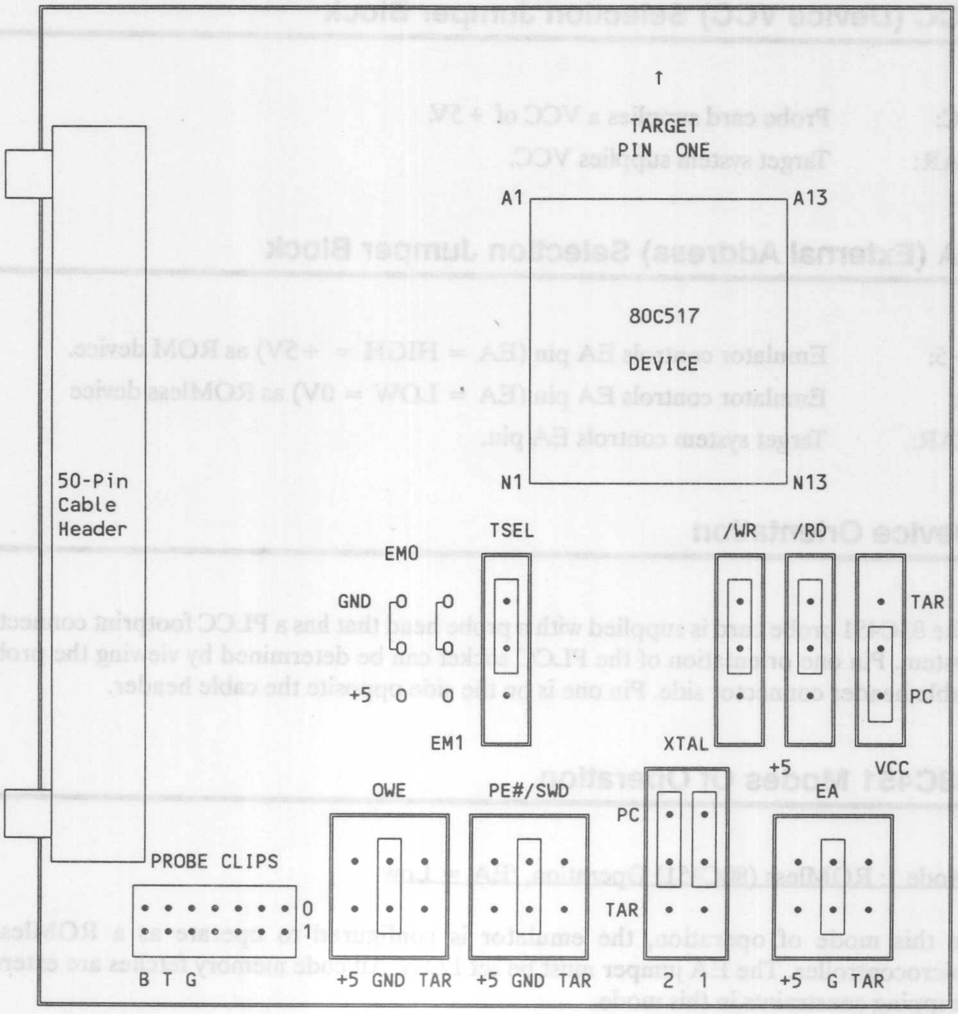
83C451 Modes Of Operation

Mode 1: ROMless (80C451) Operation, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

Mode 2: ROM (83C451), /EA = High

This is the default mode. In this mode of operation, the emulator is configured to operate as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 4K) are internal and there is no access to code memory outside of the ROM address space so there can be no external code memory fetches. The ROM address space in code memory must be mapped to the emulator in this mode.



80C517 Probe Card

TSEL (Emulation Device) Selection Jumper Block

GND: Probe card emulates 80515/80535 or 80C515/80C535 device.

The bits SD, OWDS and ADEX have no function; the oscillator watchdog is disabled; the bit WDTS is set the same way as in the 80515/80535 and 80C515/80C535; the function of pin PE#/SWD corresponds to the function of pin PE# in the 80515/80535 and 80C515/80C535 (i.e., no automatic startup of the watchdog timer). Note that the reading and writing of 80C517-only SFR's is not suppressed.

+5: Probe card emulates 80C517/80C537 device.

The bits SD, OWDS, and ADEX have their 80C517/80C537 function; the oscillator watchdog can be controlled by pin OWE (its status flag is OWDS); the function of pin PE#/SWD corresponds to the function in the 80C517/80C537.

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.

TAR: Target system supplies crystal or external clock.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

Note that if using an external clock source, XTAL2 must be driven and XTAL1 must be left unconnected.

VCC (Device VCC) Selection Jumper Block

PC: Probe card supplies a VCC of +5V.

TAR: Target system supplies VCC (power must be supplied at all times).

Note that the device on the probe card is independently bypassed at each power input regardless of where VCC is supplied from. This allows the target system to supply cleaner power for the analog to digital converter if necessary.

EA (External Address) Selection Jumper Block

+5: Emulator controls EA pin (EA = HIGH = +5V) as ROM device (80515, 80C515, 80C517).

G: Emulator controls EA pin (EA = LOW = 0V) as ROMless device (80535, 80C535, 80C537).

TAR: Target system controls EA pin.

EM0 and EM1 Jumpers

These jumpers are set at the factory to GND and must not be changed. These jumpers are reserved for possible future enhancements to the 80C517 probe card.

/RD (Read Signal) Control Jumper Block

+5: Port 3.7 is an Input/Output pin.

/RD: Port 3.7 is used as Read Signal pin.

Note that the default setting of this jumper should be /RD. However, if Port 3.7 is being used exclusively for Input/Output and you are experiencing unpredictable emulator behavior (e.g., spurious breaks, undecodable trace) change the jumper setting to +5.

/WR (Write Signal) Control Jumper Block

+5:	Port 3.6 is an Input/Output pin.
/WR:	Port 3.6 is used as Read Signal pin.

Note that the default setting of this jumper should be /WR. However, if Port 3.6 is being used exclusively for Input/Output and you are experiencing unpredictable emulator behavior (e.g., spurious breaks, undecodable trace) change the jumper setting to +5.

OWE (Oscillator Watchdog Enable) Jumper Block

+5:	OWE enabled.
GND:	OWE disabled.
TAR:	OWE controlled by target system.

PE#/SWD (Power Saving Mode Enable / Start Watchdog Timer) Jumper Block

+5:	Disable Power Saving Modes / WDT auto-start at RESET enabled (517 mode only).
GND:	Enable Power Saving Modes / WDT auto-start at RESET disabled (517 mode only).
TAR:	PE#/SWD controlled by target system.

Device Orientation

The 80C517 probe card is supplied with a probe head that has a PLCC footprint connection for the target system. Pin one side orientation of the PLCC can be determined by viewing the probe head from the cable header (component) side. Pin one is on the middle of the top side, as designated.

Functional Concerns

When emulating the 80515/80535, pin 37 on the 68 pin PLCC (MHW-CONV10) must be left open.

Note that since this is an emulation environment, the actual current (Icc) used may be greater than the maximum current specified for the device being emulated.

Note that the 80C517 probe card can be used to emulate the 80515/80535 and 80C515/80C535 devices (with a MHW-CONV10 converter) as the 80C517 is essentially a superset of those devices. Unintentional activation of 80C517-only functionality while emulating the 80515/80535 or 80C515/80C535 might lead to unexpected results. For functionality that is shared, but different, the TSEL jumper (page 4-14) controls which device to emulate.

80C517 and 80C515 (80C517 with converter) Modes Of Operation

Mode 1: ROMless (80C53X), Watchdog Timer Off, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

In addition, the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 2: ROM (80C51X), Watchdog Timer Off, /EA = High

This is the default mode. In this mode of operation, the emulator is configured as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 8K) are internal. Code memory fetches outside the ROM address space are external. The ROM area in code memory must be mapped to the emulator.

In addition, the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 3: ROMless (80C53X), Watchdog Timer On, /EA = Low

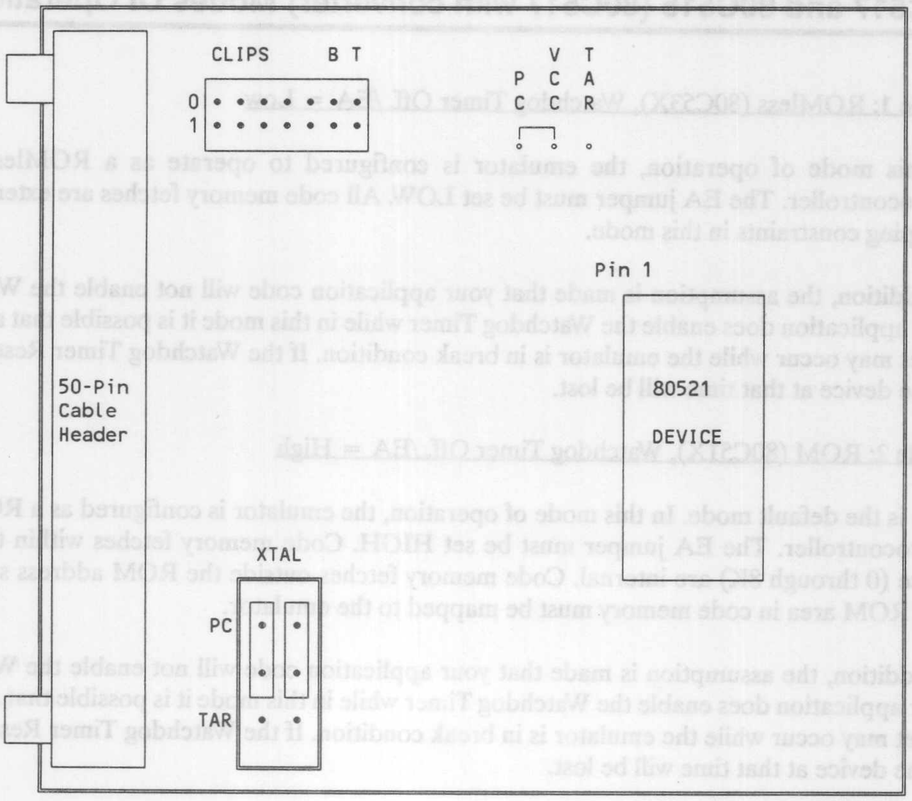
In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

In addition, the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.

Mode 4: ROM (80C51X), Watchdog Timer On, /EA = High

In this mode of operation, the emulator is configured as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 8K) are internal. Code memory fetches outside the ROM address space are external. The ROM area in code memory must be mapped to the emulator.

In addition, the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.



80C521 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.

TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

VCC (Device VCC) Jumper

The VCC jumper on this device is set at the factory to supply a VCC of +5V from the probe card. This jumper is not user-selectable and must not be changed.

Device Orientation

The 80C521 probe card is supplied with a probe head that has a DIP footprint connection for the target system. Pin one orientation of the DIP socket can be determined by viewing the probe head from the cable header (component) side. Pin one is in the upper left corner, as designated.

Devices Emulated

The 80C521 probe card will emulate only ROM devices (i.e. 80521, 8051 and 8052). This probe card will not emulate ROMless devices (i.e. 8031, 8032 and 80321). You must choose the appropriate probe card, 80C521 or 80C321, for ROM or ROMless applications, respectively.

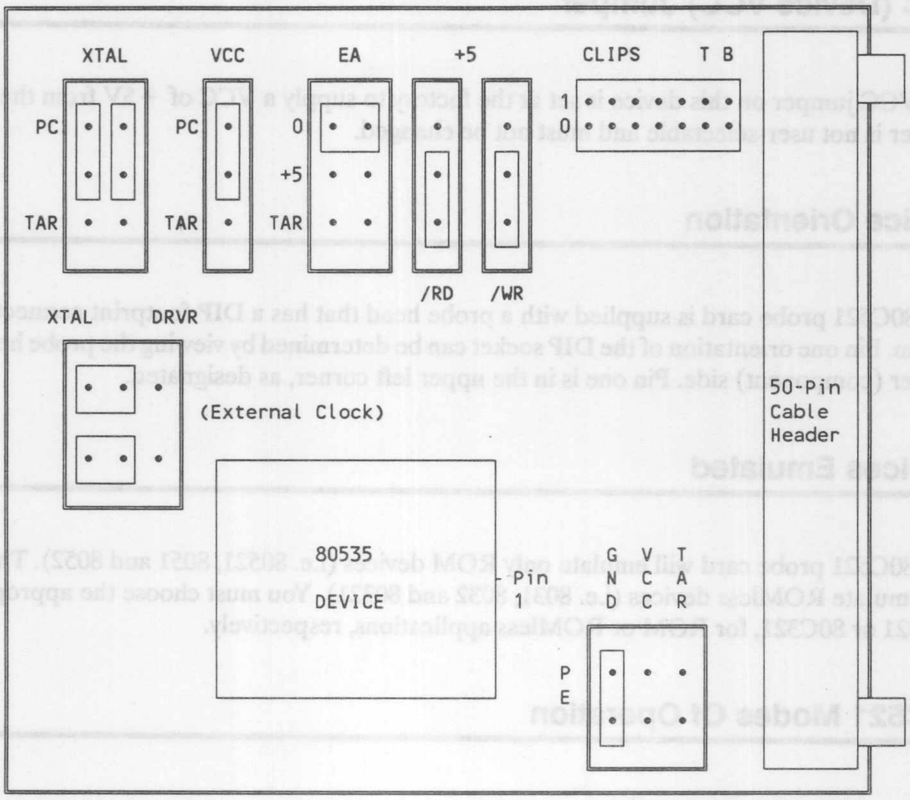
80C521 Modes Of Operation

<u>Mode 1: Watchdog Timer Off</u>

This is the default mode. In this mode of operation the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

<u>Mode 2: Watchdog Timer On</u>

In this mode of operation the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.



80C532, 80C535 Probe Cards

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.
TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

VCC (Device VCC) Selection Jumper Block

PC: Probe card supplies a VCC of +5V.

TAR: Target system supplies VCC.

EA (External Address) Selection Jumper Block

On the 80C532 and 80C535 probe card, the EA jumper block is not functional; any position may be chosen.

/RD (Read Signal) Control Jumper Block

+5: Port 3.7 is an Input/Output pin.

/RD: Port 3.7 is used as Read Signal pin.

/WR (Write Signal) Control Jumper Block

+5: Port 3.6 is an Input/Output pin.

/WR: Port 3.6 is used as Write Signal pin.

XTAL - DRVR (External Clock) Jumper Block

XTAL: Target system supplies crystal clock.

DRVR: Target system supplies clock drive.

PE (Power-down Enable) Selection Jumper Block

VCC: Emulator controls PE pin (PE = HIGH = +5V).

GND: Emulator controls PE pin (PE = LOW = 0V).

TAR: Target system controls PE pin.

Device Orientation

The 80C532 and 80C535 probe cards are supplied with a probe head that has a PLCC footprint connection for the target system. Pin one orientation of the PLCC socket can be determined by viewing the probe head from the cable header connector side. Pin one is on the side near the cable.

80C535 Modes Of Operation

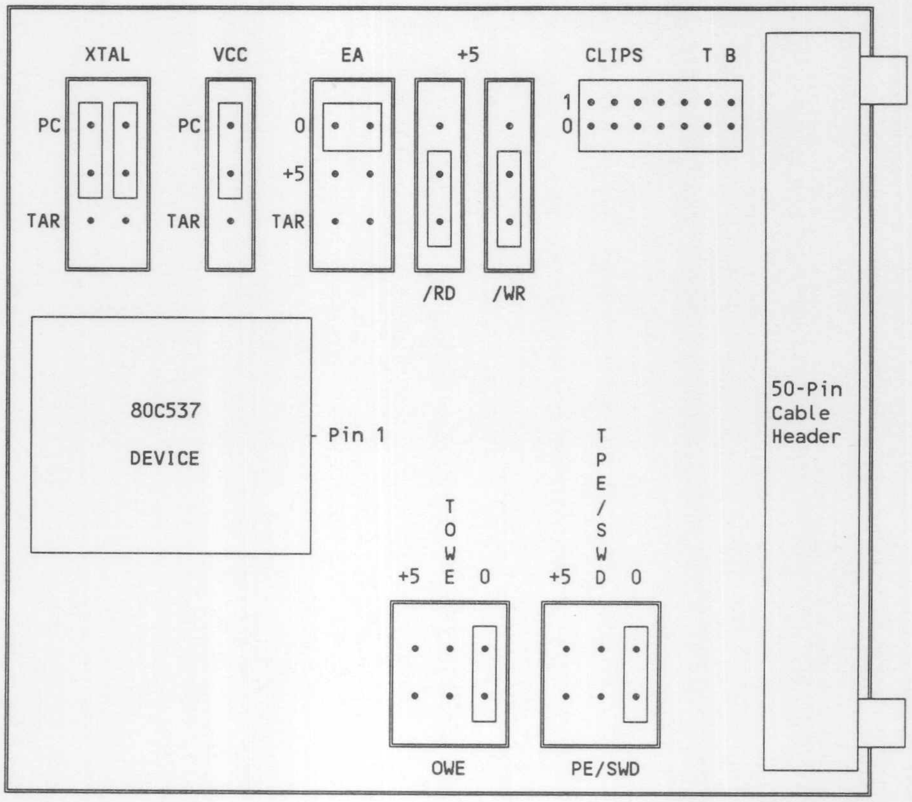
Mode 1: Watchdog Timer Off

This is the default mode. In this mode of operation the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 2: Watchdog Timer On

In this mode of operation the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.

80C537 Probe Card



80C537 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC:	Probe card's crystal is used.
TAR:	Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

VCC (Device VCC) Selection Jumper Block

PC:	Probe card supplies a VCC of +5V.
TAR:	Target system supplies VCC.

EA (External Address) Selection Jumper Block

In the 80C537 probe card, the EA jumper block is not functional; any position may be chosen.

/RD (Read Signal) Control Jumper Block

+5:	Port 3.7 is an Input/Output pin.
/RD:	Port 3.7 is used as Read Signal pin.

/WR (Write Signal) Control Jumper Block

+5:	Port 3.6 is an Input/Output pin.
/WR:	Port 3.6 is used as Write Signal pin.

OWE (Oscillator Watchdog Enable) Selection Jumper Block

+5:	Emulator controls OWE pin (OWE = HIGH = +5V).
0:	Emulator controls OWE pin (OWE = LOW = 0V).
TOWE:	Target system controls OWE pin.

PE/SWD (Power Saving Modes) Selection Jumper Block

+5:	Emulator controls PE/SWD pin (PE/SWD = HIGH = +5V).
0:	Emulator controls PE/SWD pin (PE/SWD = LOW = 0V).
TPE/SWD:	Target system controls PE/SWD pin.

Device Orientation

The 80C537 probe card is supplied with a probe head that has a PLCC footprint connection for the target system. Pin one orientation of the PLCC socket can be determined by viewing the probe head from the cable header connector side. The Pin one side is the side nearest the cable header.

80C537 Modes Of Operation

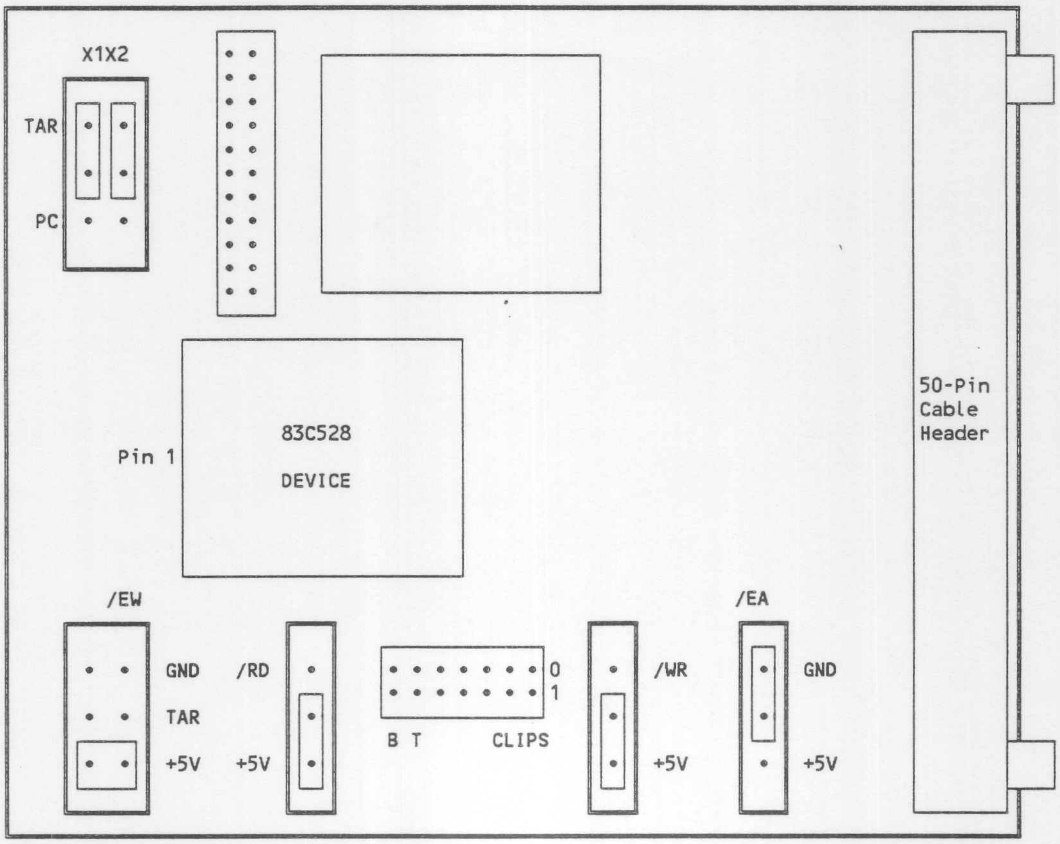
Mode 1: Watchdog Timer Off

This is the default mode. In this mode of operation the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 2: Watchdog Timer On

In this mode of operation the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.

83C528 Probe Card



83C528 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.
TAR: User's target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration.
The center post is the common post.

The LEFT side of the jumper block connects to XTAL 1 input.

The RIGHT side of the jumper block connects to XTAL 2 input.

EA (External Address) Selection Jumper Block

GND: Emulator controls EA pin (EA = LOW = 0V) as ROMless device.
+5: Emulator controls EA pin (EA =HIGH = +5V) as ROM device.

Read Signal Control Jumper Block

+5: Port 3.7 is an Input/Output pin.
/RD: Port 3.7 is used as Read Signal pin.

Write Signal Control Jumper Block

+5: Port 3.6 is an Input/Output pin.
/WR: Port 3.6 is used as Write Signal pin.

/EW (Enable Watchdog) Selection Jumper Block

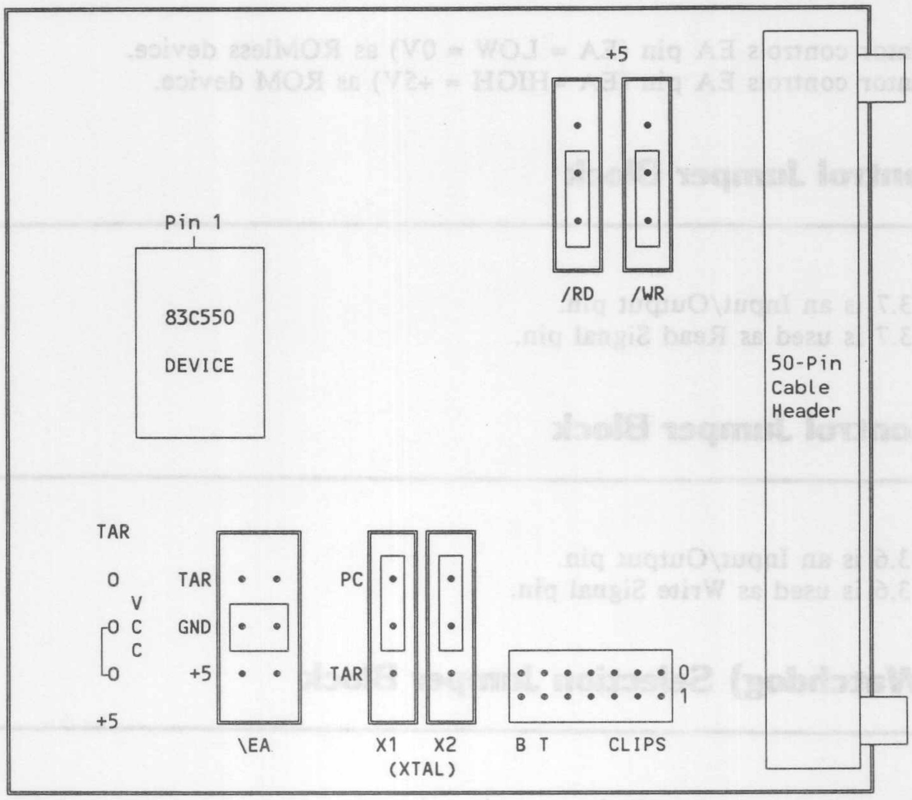
In the 83C528 probe card, the EW jumper block is not functional; any position may be chosen.

Device Orientation

The 83C528 probe card is supplied with a probe head that has a DIP or PLCC footprint connection for the target system. Pin 1 (one) orientation of the DIP or PLCC socket can be determined by viewing the probe head from the bottom (interconnect) side. Pin 1 is clearly marked on the adapter board.

NOTE

The large jumper block array next to the XTAL jumpers is factory preset and **should not** be changed.



83C550 Probe Card

X1 and X2 (XTAL--Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.

TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

X1 connects to XTAL 1 input.

X2 connects to XTAL 2 input.

Note that if using an external clock source, XTAL2 must be driven and XTAL1 must be left unconnected.

VCC (Device VCC) Jumper Block

The VCC jumper on this device is set at the factory to supply a VCC of +5V from the probe card. The jumper is not user selectable and must not be changed.

EA (External Address) Selection Jumper Block

+5:	Emulator controls EA pin (EA = HIGH = +5V) as ROM device.
G:	Emulator controls EA pin (EA = LOW = 0V) as ROMless device.
TAR:	Target system controls EA pin.

/RD (Read Signal) Control Jumper Block

+5:	Port 3.7 is an Input/Output pin.
/RD:	Port 3.7 is used as Read Signal pin.

/WR (Write Signal) Control Jumper Block

+5:	Port 3.6 is an Input/Output pin.
/WR:	Port 3.6 is used as Read Signal pin.

Device Orientation

The 83C550 probe card is supplied with a probe head that has a PLCC footprint connection for the target system. Pin one side orientation of the PLCC can be determined by viewing the probe head from the cable header (component) side. Pin one is on the middle of the top side, as designated.

Mode 1: ROMless (80C550), Watchdog Timer Off, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

In addition, the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 2: ROM (83C550), Watchdog Timer Off, /EA = High

This is the default mode. In this mode of operation, the emulator is configured to operate as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 4K) are internal and there is no access to code memory outside of the ROM address space so there can be no external code memory fetches. The ROM address space in code memory must be mapped to the emulator in this mode.

In addition, the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 3: ROMless (80C550), Watchdog Timer On, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

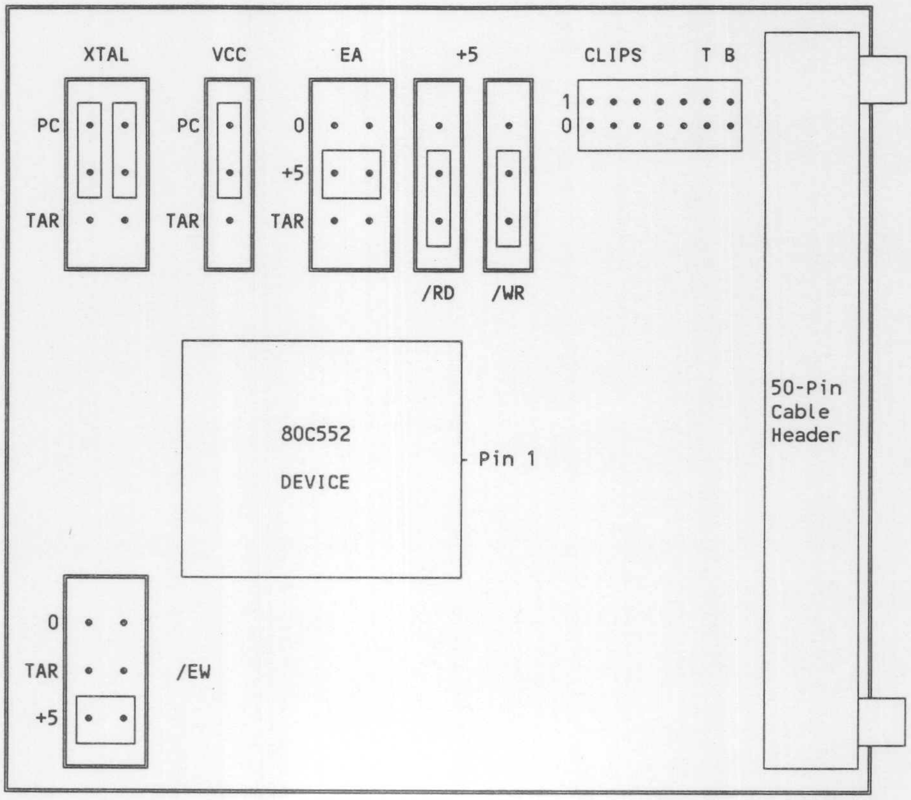
In addition, the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.

Mode 4: ROM (83C550), Watchdog Timer On, /EA = High

In this mode of operation, the emulator is configured to operate as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 4K) are internal and there is no access to code memory outside of the ROM address space so there can be no external code memory fetches. The ROM address space in code memory must be mapped to the emulator in this mode.

In addition, the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.

80C552 Probe Card



80C552 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC:	Probe card's crystal is used.
TAR:	Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

VCC (Device VCC) Selection Jumper Block

PC:	Probe card supplies a VCC of +5V.
TAR:	Target system supplies VCC.

EA (External Address) Selection Jumper Block

In the 80C552 probe card, the EA jumper block is not functional; any position may be chosen.

/RD (Read Signal) Control Jumper Block

+5:	Port 3.7 is an Input/Output pin.
/RD:	Port 3.7 is used as Read Signal pin.

/WR (Write Signal) Control Jumper Block

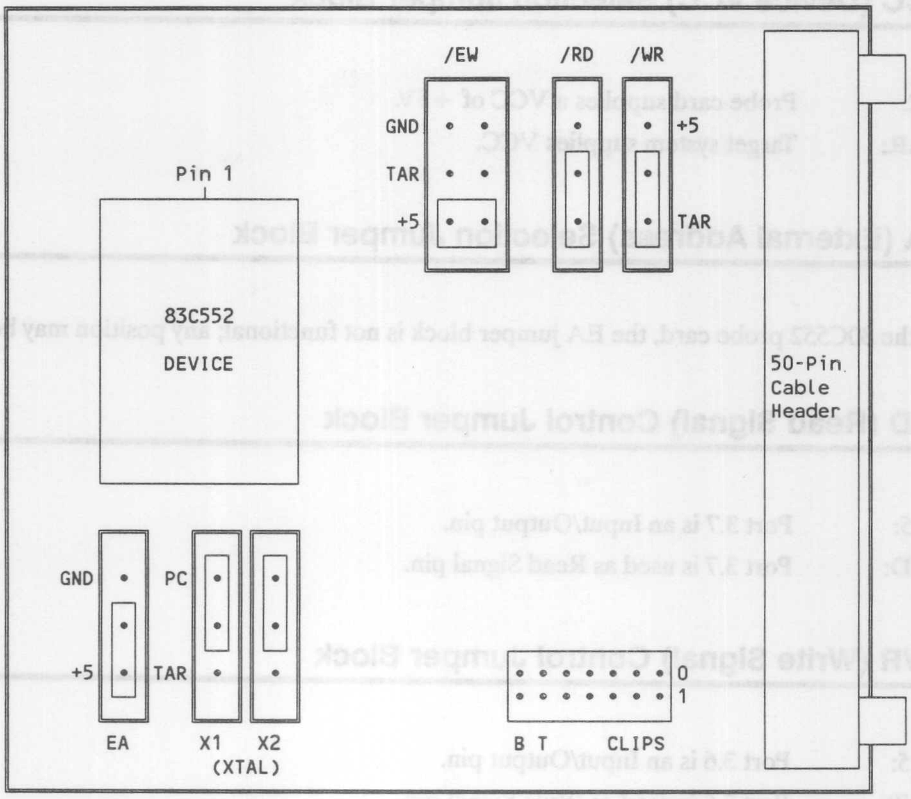
+5:	Port 3.6 is an Input/Output pin.
/WR:	Port 3.6 is used as Write Signal pin.

/EW (Enable Watchdog) Selection Jumper Block

0:	Emulator controls EW pin (EW = LOW = 0V) as Watchdog enabled.
+5:	Emulator controls EW pin (EW = HIGH = +5V) as Watchdog disabled.
TAR:	Target system controls EW pin.

Device Orientation

The 80C552 probe card is supplied with a probe head that has a PLCC footprint connection for the target system. Pin one orientation of the PLCC socket can be determined by viewing the probe head from the cable header (component) side. Pin one is on the side by the cable header on the board.



83C552, 83C652, 83C654 Probe Cards

X1 and X2 (XTAL--Oscillator) Selection Jumper Block

PC:	Probe card's crystal is used.
TAR:	Target system supplies crystal or external clock.

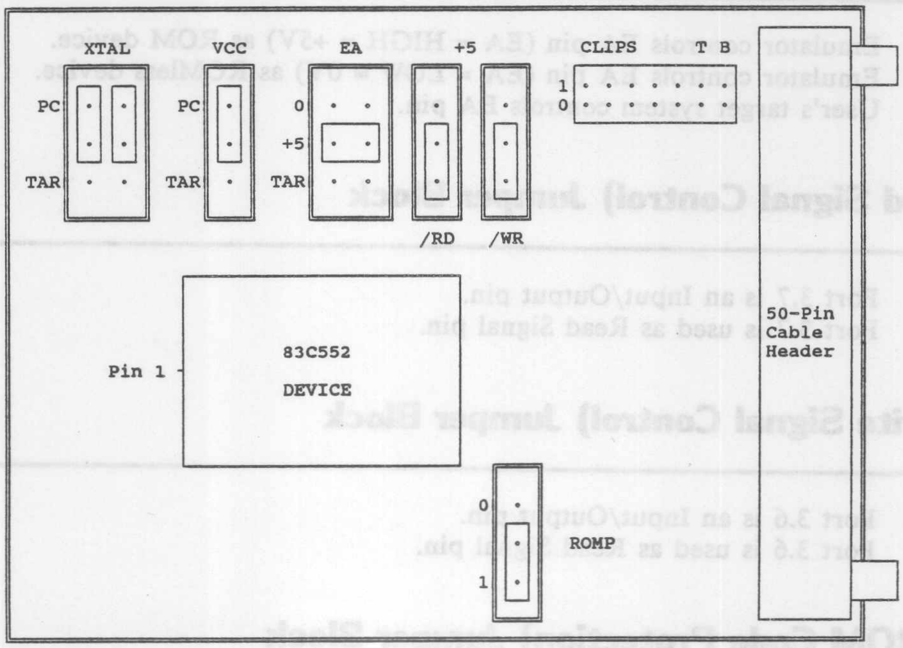
This is a double jumper to ensure correct configuration. The center post is the common post.

X1 connects to XTAL 1 input.

X2 connects to XTAL 2 input.

83C552, 83C652 and 83C654 Probe Cards (16MHz)

Revised 83C552, 83C652 and 83C654 Probe Cards. Replaces pages 4-34 and 4-35.



83C552, 83C652, 83C654 (16MHz) Probe Cards

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.
TAR: User's target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 1 input.

The RIGHT side of the jumper block connects to XTAL 2 input.

VCC (Device VCC) Selection Jumper Block

PC: Probe card supplies a VCC of +5V.
TAR: User's target system supplies VCC.

EA (External Address) Selection Jumper Block

+5: Emulator controls EA pin (EA = HIGH = +5V) as ROM device.
0: Emulator controls EA pin (EA = LOW = 0V) as ROMless device.
TAR: User's target system controls EA pin.

/RD (Read Signal Control) Jumper Block

+5: Port 3.7 is an Input/Output pin.
/RD: Port 3.7 is used as Read Signal pin.

/WR (Write Signal Control) Jumper Block

+5: Port 3.6 is an Input/Output pin.
/WR: Port 3.6 is used as Read Signal pin.

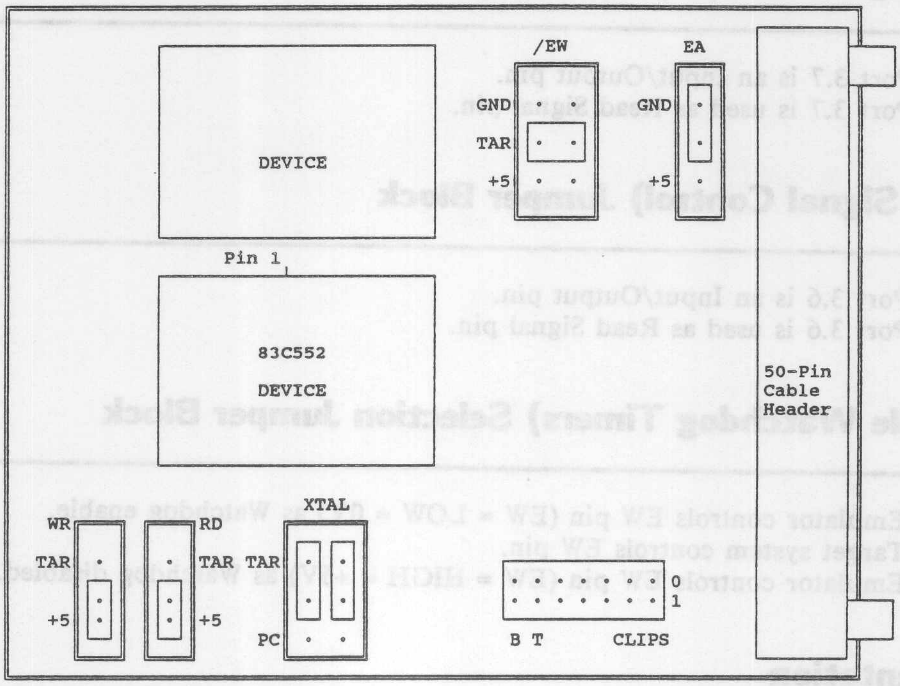
ROMP (ROM Code Protection) Jumper Block

0: ROM Code protection disabled.
No restrictions to the MOVC-instruction.

1: ROM Code protection enabled.
A MOVC-instruction executed from the external address space cannot access the internal ROM address range. There are no restrictions to the MOVC-instruction when executed from the internal address space.

Device Orientation

The 83C552, 83C652 and 83C654 probe cards are supplied with a probe head that has a PLCC footprint connection for the target system. Pin 1 orientation of the PLCC socket can be determined by viewing the probe head from the cable header side. Pin 1 is on the middle of the left side as designated.



83C552, 83C652, 83C654 (12MHz) Probe Cards

X1 & X2 (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.
TAR: User's target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

EA (External Address) Selection Jumper Block

+5: Emulator controls EA pin (EA = HIGH = +5V) as ROM device.
GND: Emulator controls EA pin (EA = LOW = 0V) as ROMless device.

RD (Read Signal Control) Jumper Block

+5: Port 3.7 is an Input/Output pin.
RD: Port 3.7 is used as Read Signal pin.

WR (Write Signal Control) Jumper Block

+5: Port 3.6 is an Input/Output pin.
WR: Port 3.6 is used as Read Signal pin.

/EW (Enable Watchdog Timers) Selection Jumper Block

GND: Emulator controls EW pin (EW = LOW = 0V) as Watchdog enable.
TAR: Target system controls EW pin.
+5: Emulator controls EW pin (EW = HIGH = +5V) as Watchdog disabled.

Device Orientation

The 83C552, 83C652 and 83C654 probe cards are supplied with a probe head that has a PLCC footprint connection for the target system. Pin 1 orientation of the PLCC socket can be determined by viewing the probe head from the cable header side. Pin 1 is on the top of the 87C552 socket device side as designated.

EA (External Address) Selection Jumper Block

GND:	Emulator controls EA pin (EA = LOW = 0V) as ROMless device.
+5:	Emulator Controls EA pin (EA = HIGH = +5V) as ROM device.

/RD (Read Signal) Control Jumper Block

+5:	Port 3.7 is an Input/Output pin
TAR:	Port 3.7 is used as Read Signal pin.

/WR (Write Signal) Control Jumper Block

+5:	Port 3.6 is an Input/Output pin.
TAR:	Port 3.6 is used as Write Signal pin.

/EW (Enable Watchdog) Selection Jumper Block

GND:	Emulator controls EW pin (EW = LOW =0V) as Watchdog enabled.
+5:	Emulator controls EW pin (EW = HIGH = +5V) as Watchdog disabled.
TAR:	Target system controls EW pin.

Device Orientation

The 83C552, 83C652 and 83C654 probe cards are supplied with a probe head that has a PLCC footprint connection for the target system. Pin 1 orientation of the PLCC socket can be determined by viewing the probe head from the cable header connector side. Pin one is on the middle of the top side.

83C552 Modes Of Operation

Mode 1: ROMless (80C552), Watchdog Timer Off, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

In addition, the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 2: ROM (83C552), Watchdog Timer Off, /EA = High

This is the default mode. In this mode of operation, the emulator is configured as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 8K) are internal. Code memory fetches outside the ROM address space are external. The ROM area in code memory must be mapped to the emulator.

In addition, the assumption is made that your application code will not enable the Watchdog Timer. If your application does enable the Watchdog Timer while in this mode it is possible that a Watchdog Timer Reset may occur while the emulator is in break condition. If the Watchdog Timer Reset occurs the state of the device at that time will be lost.

Mode 3: ROMless (80C552), Watchdog Timer On, /EA = Low

In this mode of operation, the emulator is configured to operate as a ROMless version of the microcontroller. The EA jumper must be set LOW. All code memory fetches are external. There are no mapping constraints in this mode.

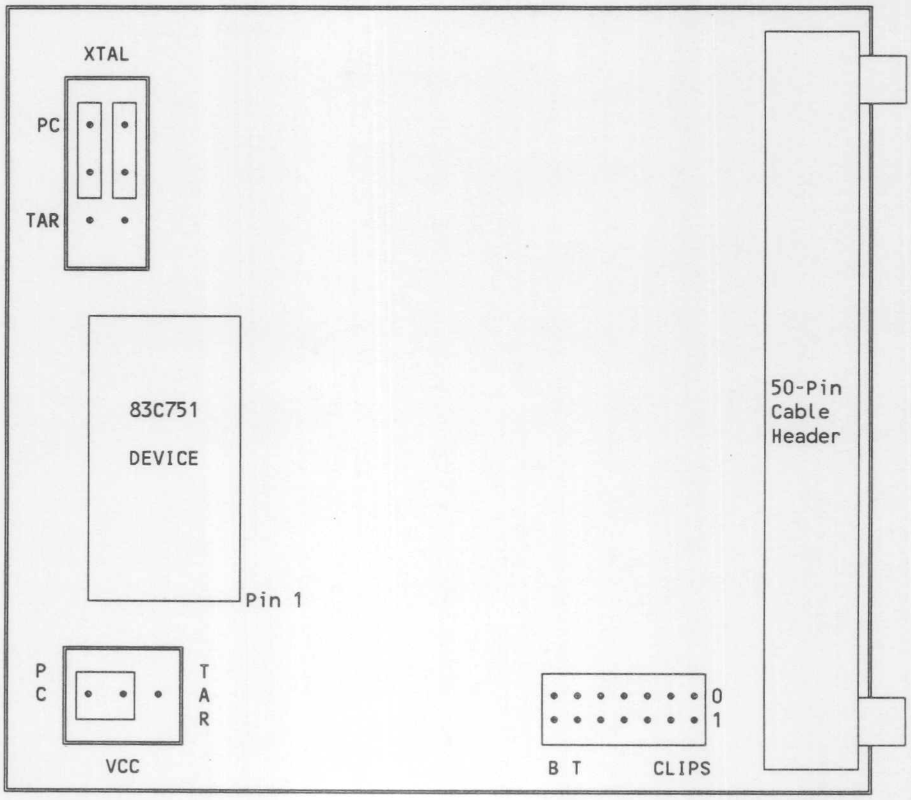
In addition, the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.

Mode 4: ROM (83C552), Watchdog Timer On, /EA = High

In this mode of operation, the emulator is configured as a ROM version of the microcontroller. The EA jumper must be set HIGH. Code memory fetches within the ROM address space (0 through 8K) are internal. Code memory fetches outside the ROM address space are external. The ROM area in code memory must be mapped to the emulator.

In addition, the assumption is made that your application code has enabled the Watchdog Timer. While in break condition in this mode, the emulator executes the code necessary to refresh the Watchdog Timer to prevent a Watchdog Timer Reset from occurring. Of course, while in emulation condition your application code is responsible for refreshing the Watchdog Timer.

83C751 Probe Card



83C751 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC:	Probe card's crystal is used.
TAR:	Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

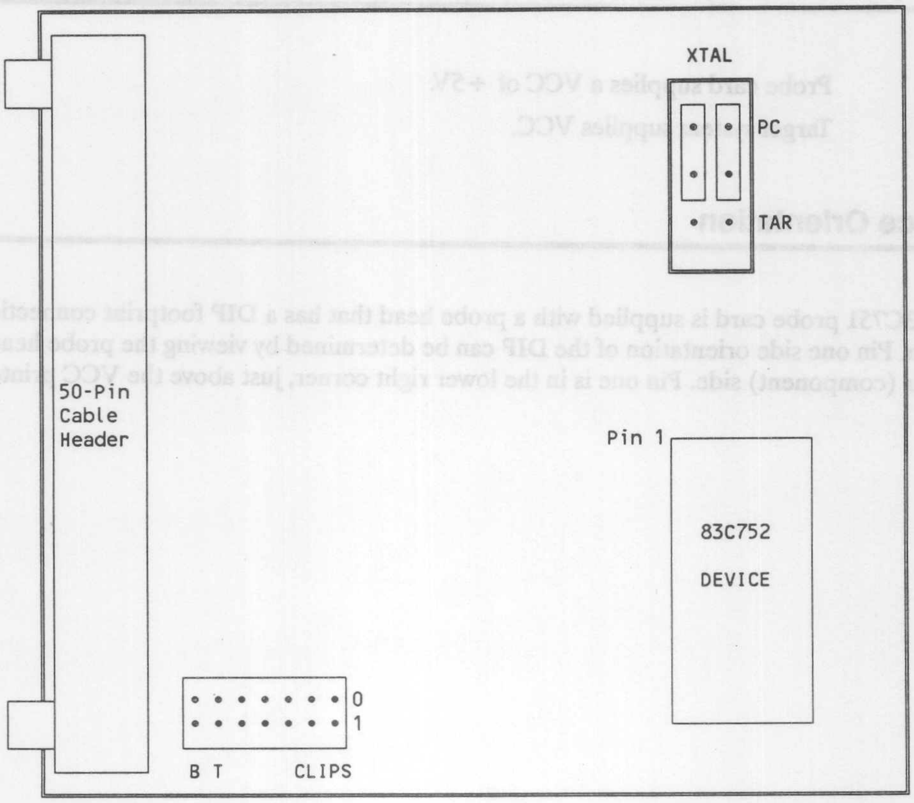
The RIGHT side of the jumper block connects to XTAL 1 input.

VCC (Device VCC) Selection Jumper Block

PC:	Probe card supplies a VCC of +5V.
TAR:	Target system supplies VCC.

Device Orientation

The 83C751 probe card is supplied with a probe head that has a DIP footprint connection for the target system. Pin one side orientation of the DIP can be determined by viewing the probe head from the cable header (component) side. Pin one is in the lower right corner, just above the VCC printed on the board.



83C752 Probe Card

XTAL (Oscillator) Selection Jumper Block

PC: Probe card's crystal is used.

TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

The LEFT side of the jumper block connects to XTAL 2 input.

The RIGHT side of the jumper block connects to XTAL 1 input.

Device Orientation

The 83C752 probe card is supplied with a probe head that has a DIP footprint connection for the target system. Pin one side orientation of the DIP can be determined by viewing the probe head from the cable header (component) side. Pin one is in the upper left corner of the device.

Map of Clock Jumpers for 803X Type Probe Cards

Map of Clock Jumpers for iceMASTER 8031/8032/8344/80C51FA/80C154/80C321/80C410/80C652/80C851								
			Clock Source					
			Crystal			Jumpers		
Probe Card	Device Type On Probe Card	Target Device Type *	Probe Card	Target System	External Clock Driver	XTAL	C/N-NORM	XTAL1-XTAL2
8344	NMOS	NMOS	X			PC	NORM	XTAL2
8344	NMOS	NMOS		X		TAR	NORM	XTAL2
8344	NMOS	NMOS			X	TAR	NORM	XTAL2
**	CMOS	CMOS	X			PC	NORM	XTAL2
**	CMOS	CMOS		X		TAR	NORM	XTAL2
**	CMOS	CMOS			X	TAR	NORM	XTAL2
**	CMOS	NMOS	X			PC	NORM	XTAL2
**	CMOS	NMOS		X		TAR	NORM	XTAL2
**	CMOS	NMOS			X	TAR	C/N+	XTAL2

* Under the "Target Device Type" column, CMOS means any CMOS microcontroller supported by the iceMASTER emulator base. NMOS means any NMOS microcontroller supported by the iceMASTER emulator base.

** These marked positions apply to all iceMASTER Probe Cards except the 8344.

+ C/N means "CMOS part emulating an NMOS part".

Map of Clock Jumpers for the 8052 Probe Card

Map of Clock Jumpers for iceMASTER						
		Clock Source			Jumpers	
		Crystal		External Clock Driver		
Probe Card	Target Device Type	Probe Card	Target System		XTAL	CMOS NMOS
8052	CMOS	X			PC	CMOS
8052	CMOS		X		TAR	CMOS
8052	CMOS			X	TAR	CMOS
8052	NMOS	X			PC	CMOS
8052	NMOS		X		TAR	CMOS
8052	NMOS			X	TAR	NMOS

Emulator System Requirements

Be sure you have everything you need. A basic emulator system includes a power supply, the iceMASTER emulator base, one or more probe cards (depending on how many you ordered), one 50 pin flat cable, one RS-232 serial cable with a 25 pin connector, one probe clip assembly, one manual and two software diskettes.

A power source is required to operate the emulator. You may use your own power supply (5 volts at 1.5 amperes is required) or you may purchase a MetaLink optional power supply.

Please stop now and take a moment to be sure you have all the items you need. If you are missing something, call us!

Host Computer Requirements

At a minimum, the Host Computer requirements are:

1) IBM PC/XT/AT or 100% compatible,
2) 640K bytes or RAM,
3) one floppy disk drive,
4) one hard disk dive,
5) one serial port, and
6) DOS 2.0 (or later).

You may perform the software installation before or after the hardware installation. Both must be completed, however, before you can use either.

Installation

The emulator Host Software system is shipped on two 5-1/4 inch, double-sided, double-density floppy diskettes (unless otherwise requested). One diskette is labeled **Host Software** and the other is labeled **Probe Card Software.** Be sure to make backup copies of these diskettes, never remove the write-protect tabs from either diskette and store the diskettes in a safe place. The files on both diskettes are in a packed format and are self-unpacking.

The installation procedure will unpack the files from the distribution diskettes, creating the Host Software system, utility and example program files on your hard disk. This will require approximately 2.1 megabytes of hard disk space before beginning installation. After installation the Host Software system will occupy approximately 1.7 megabytes of hard disk space.

To start the software installation process, place the diskette labeled **Host Software** in drive A: and type

A:INSTALL

The install program will lead you through the installation procedure. The purpose of each unpacked file is explained in the README_1.DOC and README_2.DOC files. These files are displayed as part of the installation procedure. In addition, the utility program MF_GEN.EXE is invoked automatically to generate a $MODEL file to allow use of the emulator probe card that was purchased. This $MODEL file is read by the Host Software during its initialization sequence. Note that MF_GEN.EXE may be reinvoked manually to generate a $MODEL file for another probe card. If you do this be sure to rename it, or use the name definition ability in MF_GEN.EXE to name the new file something other than the default, which is $MODEL, to avoid overwriting the existing $MODEL file.

This chapter is an overview of some of the features of the Host Software and a guide to help you understand the layout of the screen and the terminology used to describe the Host Software. For complete information on any command please refer to the Command Reference (Chapter 7).

Terminology

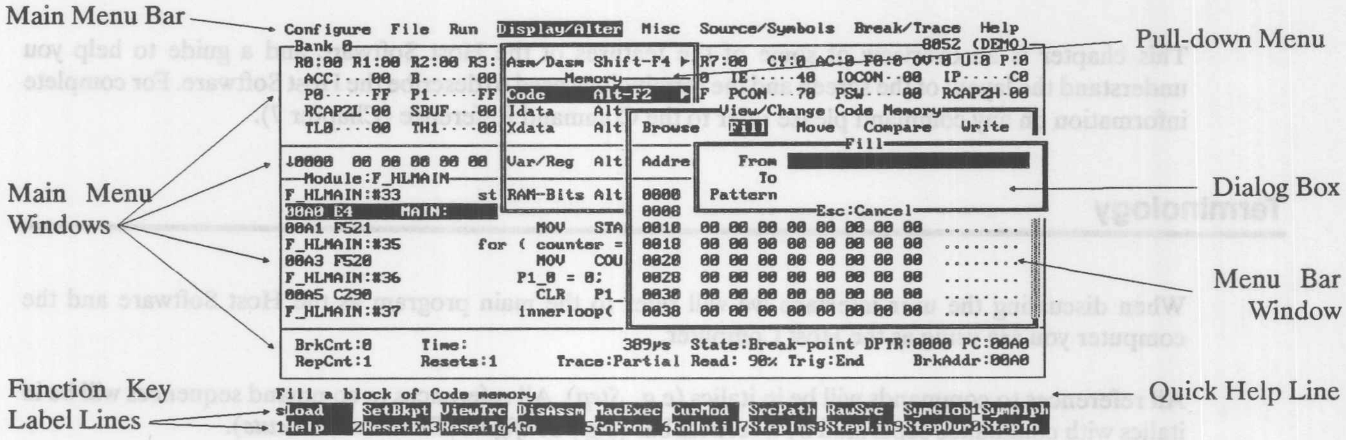
When discussing the user interface we will refer to the main program as the Host Software and the computer you are using as the Host Computer.

All references to commands will be in italics (e.g., *Step*). All references to command sequences will be in italics with commands separated by a vertical bar (e.g., *Configure|Emulator|Execute*).

References to file names, directory structures and disk drive designations will always be shown in upper case (e.g., C:\IM51\DEMO.DBG).

Where shown, user responses will be in boldface and special keys will be shown in boldface as they are depicted on a standard IBM PC keyboard (where possible), as follows:

Alt Key	**Alt**
Arrow or Cursor Keys	**← → ↑ ↓**
Control Key	**Ctrl**
Delete Character To Left Key	**Bksp**
Delete Key	**Del**
End Key	**End**
Enter or Carriage Return Key	**Enter**
Escape Key	**Esc**
Function Keys	**F1 ... F10**
Home Key	**Home**
Insert Key	**Ins**
Page Down Key	**PgDn**
Page Up Key	**PgUp**
Print Screen Key	**PrtSc**
Space Bar	**Space**
Tab Key	**Tab**



Quick Help Line The **Quick Help Line** refers to the line on the display just above the Function Key Label Lines. If none are displayed then the Quick Help Line will be the bottom line of the display. The Quick Help Line is used to explain (briefly) what the currently highlighted command is used for. This line is always displayed.

Function Key Label Lines **Function Key Label Lines** are optionally displayed at the bottom of the display, below the Quick Help Line. They are used as a quick reference to the 40 Function Key Assignments (Hot Keys). By default 2 lines are displayed but up to 4 lines can be displayed (see *Configure | Function Key | Lines* on page 7-23).

Window Note that the term **Window** is a descriptive term that refers to various parts of the display presented by the Host Software such as the RAM-bits Window (see *Display/Alter | RAM-Bits* on page 7-40). It is not intended to imply compatibility with the Microsoft Windows operating environment or any other similar system.

Main Screen Windows The **Main Screen Windows** are a special subset of all windows. They are intended to be the platform from which an emulation session is controlled. Access to all the major groups of commands is available from the Main Menu Bar. The Main Screen Windows are completely configurable for content and position (see *Configure | Windows | Modify* on page 7-17) and for size (see *Configure | Windows | Size* on page 7-18). See the *Configure | Windows* command on page 7-13 for a complete description of the window types available on the Main Screen. Note that the data in all of the Main Screen Windows is updated every time there is a break from emulation, including after a single step (see *Run | Step* on page 7-34 and between each instruction during slow motion mode (see *Run | Slow Motion* on page 7-34) and whenever the *Configure | Windows | Repaint* command (page 7-19) is selected.

Menu Bar Window A **Menu Bar Window** is a window that contains a selection of commands arranged horizontally. The highlighted command is the current command. When a command is selected (see Selecting Commands on page 6-4 in this chapter) it may call another Menu Bar Window, it may call a Pull-down Menu, or it may perform a function.

Pull-down Menu A **Pull-down Menu** contains a selection of commands (generally related) arranged vertically in a box positioned (if possible) just below the command that was selected. The highlighted command is the current command. Just as in a Menu Bar, when a command is selected it may call a Menu Bar Window or a Pull-down Menu, or perform a function.

Dialog Box A **Dialog Box** refers to a box that is presented on the display to prompt you for some information such as a filename (see *File | Load* on page 7-27) or an address (see *Run | Until* on page 7-34). Most Dialog Boxes retain the information you typed from the last time the same command was selected. Editing of retained and newly typed information is possible using the following keys:

←	Move cursor one position to left
→	Move cursor one position to right
Ctrl-Home	Move cursor to left most position of line
Ctrl-End	Move cursor to right most position of line
Ins	Toggle insert/overwrite mode
Del	Delete character under cursor
Bksp	Delete character to the left of cursor and move cursor one position to left
Enter	Accept typed (or edited) information
Esc	Cancel input

Note that the Dialog Box used to prompt for filenames is a special form of Dialog Box that includes all the standard features mentioned above, plus the ability to view directory listings. See the description of the Filenames and Directory Facility features in this chapter.

Diagnostic Message Box A **Diagnostic Message Box** refers to a box that is presented to display error and warning messages and other information important enough for you to see. Generally, the Diagnostic Message Box will be presented in the center of the display and the computer will beep, unless this feature is disabled (see *Configure | Options | Diagnostic* on page 7-21). The Diagnostic Message Box will remain on the display until you press **Esc**.

Status Box **Status Boxes** are generally used to inform you of the progress of a task that the Host Software is performing and require no response from you. For example, when loading a file several Status Boxes will be displayed to inform you of the progress of the file load. Note that at times the Status Boxes may be removed from the screen before you can read them. This merely indicates that the task being performed took only a short time to complete.

Confirmation Box A **Confirmation Box** refers to a box that is presented to confirm your intention to execute the command you selected. All Confirmation Boxes accept only a Yes (Y), No (N) and Cancel (C or Esc) response. Note that in most cases the No and Cancel response are equal, but there are several cases where they are not (e.g., *Display/Alter | Code | Write | Dump* on page 7-39). In any case, the Confirmation Box will tell you what each option means where it is not obvious.

Selecting Commands

Host Software commands may be selected using the keyboard or a mouse (see Appendix I for information on using a mouse). Selecting a command is the generic phrase used to describe the process of executing a command by moving the highlight bar (via cursor keys or mouse) to the desired command and pressing **Enter** or using a Quick Key or a Hot Key.

Quick Key refers to the highlighted alphanumeric character in a command name which can be used to select a command. The Quick Key is most often the first character in the command name. For example, when working at the Main Menu Bar, pressing the letter C means select the *Configure* command.

Hot Key refers to one of the 40 Function Keys that can be used to select a command, as follows:

F1	...	**F10**
Shift-F1	...	**Shift-F10**
Ctrl-F1	...	**Ctrl-F10**
Alt-F1	...	**Alt-F10**

The Function Keys are assigned at the factory to commonly used commands (see Appendix N for a list of default assignments) but can be reassigned using the *Configure|Function Key|Modify* command (page 7-23). Note that a Hot Key can be used from just about anywhere in the Host Software (except during an emulation or a performance analysis) to immediately select a command without having to traverse the menu structure to find the command.

Features

Help

Detailed and context sensitive **Help** is available on-line using the Hot Key assigned to Help (by default **F1**, but this can be changed using the *Configure|Function Key|Modify* command on page 7-23). Pressing the Help Hot Key pops up a Help Window to display information about the currently highlighted command. Once you are in the Help System, information may also Hyperlinked to other, related Help topics.

Command Line Options

There are several command line options that provide useful shortcuts for entering the Host Software and performing tasks that would require executing several commands. The following are examples of some of these shortcuts:

ICE DEMO.DBG	Load DEMO.DBG (see *File|Load* on page 7-27) and establish communication with the emulator (see *Configure|Emulator* on page 7-2).
ICE -S C:\	Set the HLL search path to c:\ (see *Source/Symbols|Source Path* on page 7-52).
ICE -I DEMO.MAC	Start the macro file DEMO.MAC (see *File|Macro* on page 7-30).

There are many command line options, each of which is described in detail in Appendix J.

Macros

Macros allow you to create scripts of keystrokes (that may be commands or responses to prompts) that can be called when needed. Typically, these would be for repetitive tasks and tasks such as setup for a debug session, but can be used for any purpose. For a full description of Macros refer to *File|Macro* on page 7-30.

Warm Start If an emulator is powered up and configured, you may leave the Host Software to return to DOS and then return to the emulator Host Software at a later time without having to reconfigure the emulator. All symbolic information and map settings will be lost but the emulator's memory segments (code, external data, internal data, and SFR's) will retain their values. Of course, this requires that power to the emulator be maintained between the debugging sessions.

Dynamically Annotated Code When emulating using single steps (*Run*|*Step* on page 7-34) or slow motion mode (*Run*|*Slow Motion*) on page 7-34), the right side of the Main Source Window is used to display a history of execution for each instruction executed. This history contains the value (before execution of the instruction) of any Direct Address, Indirect Address, SFR and Bit used by the instruction. In addition, if the instruction is an unconditional jump instruction or a conditional jump, where the condition is met (TRUE), the Target Address and an arrow indicating direction of program flow will be displayed. If the instruction is a conditional jump, where the condition is not met (FALSE), a * is displayed.

Moving A Window Many windows can be moved to any position on the display using the **Ctrl-V** command. Once you press the **Ctrl-V** key combination, a new border will be painted around the current window and the window can then be moved. The computer will beep when the window cannot be moved in the direction specified (e.g., a window cannot cover the Quick Help Line).

At this point the following keys are active:

← → ↑ ↓	move whole window in indicated direction
Esc	abort move, leave the window in the position it was in when **Ctrl-B** was pressed
Enter	exit move, accept new window position

The Main Screen Windows are moved in another way. Their ordering and position may be changed using the *Configure*|*Windows*|*Modify* command (page 7-17). Note that for the Main Screen Windows using the **Ctrl-V** command is equivalent to selecting the *Configure*|*Windows*|*Modify* command.

Resizing A Window Many windows can be resized on the display using the **Ctrl-R** command. Once you press the **Ctrl-R** key combination, a new border will be painted around the current window and the window can then be resized. The computer will beep when the window cannot be resized in the direction specified.

At this point the following keys are active:

↑ ↓	move bottom border in indicated direction
← →	move right border in indicated direction
Esc	abort resize, leave the window the size it was in when **Ctrl-R** was pressed
Enter	exit resize, accept new window size

The Main Screen Windows are resized in a similar fashion. Their size may be changed using the *Configure*|*Windows*|*Size* command (page 7-18). Note that for the Main Screen Windows using the **Ctrl-R** command is equivalent to selecting the *Configure*|*Windows*|*Size* command.

Change Mode When in Change Mode you may use the cursor keys to select a data item in the current window and modify its value. Change Mode is available from the *Display/Alter|Code* Window (page 7-38), the *Display/Alter|Idata* Window (page 7-38), the *Display/Alter|Xdata* Window (page 7-38) and for all Main Screen Windows except the Main Source Window. In the Main Register Window, Change Mode is indicated by a highlighted data item and in other windows a box containing the address of the data item will be displayed underneath the highlighted data item.

To enter Change Mode from the *Display/Alter|Code* Window, the *Display/Alter|Idata* Window and the *Display/Alter|Xdata* Window you must first be in Browse Mode (see *Display/Alter|Code|Browse* on page 7-38). When in Browse Mode press **Tab** or **Shift-Tab** (they are equivalent) to enter Change Mode.

To enter Change Mode from the Main Screen Windows press **Tab** or **Shift-Tab** (which is simply a convenient shortcut for the *Configure|Windows|Goto* command on page 7-18). The first time either key is pressed Change Mode becomes active in the window at the top of the display. Subsequent **Tabs** make Change Mode active in the next window down, each time advancing one window and wrapping from the bottom window back to the top window. Subsequent **Shift-Tabs** make Change Mode active in the next window up, each time moving up one window and wrapping from the top window back to the bottom window. Note that windows (i.e., Main Source) where Change Mode is unavailable are skipped during this process.

Once in Change Mode the following keys are active:

← → ↑ ↓	move currently selected data item
PgUp	move up one window full of data
PgDn	move down one window full of data
Home	move to beginning of data
End	move to end of data
Esc	exit Change Mode
Enter	open a Dialog Box to enter new data
Tab	Main Windows: next window Display/Alter: exit Change Mode
Shift-Tab	Main Windows: previous window Display/Alter: exit Change Mode

Note also that pressing any alphanumeric character while in Change Mode will open a Dialog Box to enter new data.

Filenames

Whenever the Host Software needs a filename, a special Dialog Box is used to prompt for the name. When prompted for a filename, you may request a directory listing (explained below), or may enter a file specification. A file specification is any legal DOS file specification which may include a drive designation and directories in addition to the filename. Some legal DOS file specifications follow:

DEMO.DBG	DEMO.DBG in default directory on default drive
C:DEMO.DBG	DEMO.DBG in default directory on drive C:
C:\IM51DEMO\DEMO.DBG	DEMO.DBG in \IM51DEMO directory on drive C:

After entering a file specification the Host Software will attempt to open the file.

Reading A File If the file cannot be found or cannot be opened you will be notified by an error message, as follows:

File 'filename' not found. ("DOS message")

where "DOS message" explains the DOS file open error code (explained in the DOS reference manual). If the file is found and can be opened, informational messages will be displayed to inform you of the progress of the read operation. These messages will vary depending on what type of file is being read.

Writing A File If the file cannot be found the Host Software will automatically create the file. If the file cannot be opened you will be notified by an error message, as follows:

Cannot open 'filename'. ("DOS message")

where "DOS message" explains the DOS file open error code (explained in the DOS reference manual).

If the file already exists and the information to be written is intended as human readable text (such as a write of the decoded trace buffer) you may want to append this information to the existing file. In this case you will be prompted through a Confirmation Box as follows:

File already exists. Do you wish to append? (Yes = append, No = overwrite)

where pressing **Y** means append (Yes) to the end of the existing file, pressing **N** means overwrite (No) the existing file and pressing **Esc** or **C** cancels the write operation.

If the file already exists and the information to be written is intended as reloadable data (such as a write of code memory in hex format) you may not want to overwrite the existing file. In this case you will be prompted through a Confirmation Box as follows:

File already exists. Do you wish to overwrite?

where pressing **Y** means overwrite (Yes) the existing file and pressing **N**, **Esc** or **C** cancels the write operation.

At any filename prompt you may obtain a directory listing by pressing either ? or !. Pressing ? invokes a terse (filename only) directory listing (the same information produced by the DOS command **DIR /W/P**). Selecting ! invokes a verbose (filename, size, and date) directory listing (the same information produced by the DOS command **DIR /P**).

After choosing either type of directory listing a Dialog Box will prompt you for the directory to list. You may enter any legal DOS path specification which may include a drive and/or directories. Just pressing **Enter** selects a directory listing of the default directory on the default drive. Some legal DOS path specifications follow:

C:	default directory on drive C:
\IM51	\IM51 directory on default drive
A:\IM51\EXAMPLES	\IM51\EXAMPLES directory on drive A:

The directory listing will be displayed a page at a time. Each page will be displayed until the user responds to the prompt:

Press any key to continue.

In order to use the directory facility the Host Software must have access to the COMMAND.COM file. See DOS Information (Appendix L) for detailed information on the COMMAND.COM file.

Chapter 7: Command Reference

The Main Menu Bar lists seven major groups of commands (plus *Help*). Related commands and functions of the Host Software are gathered together under these headings. This chapter will discuss each command in-depth in the left-to-right and top-to-bottom sequence in which you will encounter them in the Host Software.

As we proceed, each subpath encountered will be taken until there are no further possible branches. Discussion will then continue with the next command in sequence. Note that since different devices have different features, some commands will be unavailable for use. These commands will be de-activated in the Host Software and will use the de-emphasized display attributes assigned in the current video settings.

The major command groups are:

Configure

File

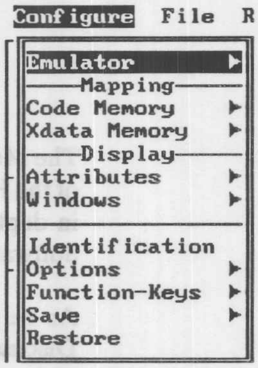
Run

Display/Alter

Misc

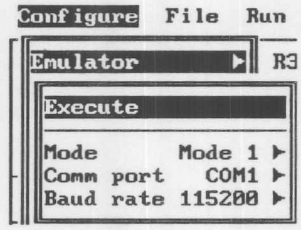
Source/Symbols

Break/Trace



The *Configure* command calls a Pull-down Menu from which you have access to commands that enable you to configure your emulation environment, including both Hardware (communication parameters, operating mode and mapping) and Software (video parameters, window configuration and Function Key assignment). The commands are:

Configure|Emulator



The *Emulator* command calls a Pull-down Menu from which you may set the operating mode of the microcontroller and communication parameters, and establish communication with the emulator. The current settings for each are displayed in the Pull-down Menu next to the command associated with that function. The commands are:

Configure|Emulator|Execute

The *Execute* command causes the Host Software to establish communication with the emulator base and sets the operating mode of the microcontroller.

The communication parameters (communication port and baud rate) and operating mode used are read from the $CONFIG file, if it exists. If the $CONFIG file does not exist, default values are used. In either case, the current setting for each of the values is shown in the Pull-down Menu. The defaults are:

Mode	device dependent (see Probe Card Reference)
Comm port	COM1
Baud rate	115200

Configure|Emulator|Mode

If the *Mode* command is active that means there is more than one possible mode of operation for the current device. Selecting the *Mode* command calls a Pull-down Menu from which you may select the desired mode of operation for your environment. If the *Mode* command is de-activated that means there is only one possible mode of operation for your device (e.g., the 8032 is a ROMless only device). The default mode of operation is device dependent, therefore see the Probe Card Reference for Mode information on your device.

Note that if the mode of operation is changed the $CONFIG file is updated (or created) automatically but the actual mode of operation of the device will not be set until the *Execute* command is selected.

Configure | Emulator | Comm port

The *Comm port* command calls a Pull-down Menu from which you may select the active communication port used for serial communication between the Host Computer and the emulator base. The default communication port used is *COM1*.

Note that if the communication port is changed the $CONFIG file is updated (or created) automatically but the actual communication port used during serial communication will not be updated until the *Execute* command is selected.

Configure | Emulator | Baud rate

The *Baud rate* command calls a Pull-down Menu from which you may select the baud rate used during serial communication between the Host Computer and the emulator base. There are six possible values for Baud Rate:

115200

57600

38400

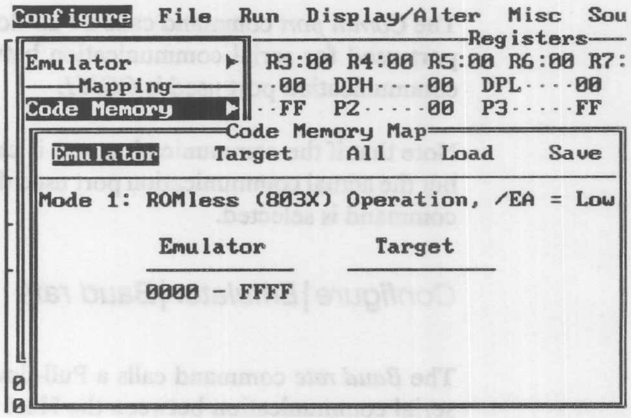
28800

19200

9600

Setting the baud rate is handled by the Host Software and is independent of any baud rate set by you from DOS. The default baud rate used is 115200.

Note that if the baud rate is changed the $CONFIG file is updated (or created) automatically but the actual baud rate used during serial communication will not be updated until the *Execute* command is selected.



The *Code Memory* command calls a Menu Bar Window from which you may map Code Memory to either the emulator's Code Memory or your target board's Code Memory in 16-byte blocks. Changes in map settings are downloaded to the emulator as they are made.

Note that there are some mapping constraints due to emulator and device dependencies (e.g., in ROM mode all ROM space must be mapped to the emulator). The mapping constraints are enforced. Mapping constraints (if any) are discussed in the description of each probe card in the Probe Card Reference (Chapter 4). The current mode of operation of your device is displayed in the window. The commands are:

Configure | Code Memory | Emulator

The *Emulator* command allows you to map blocks of Code Memory to the emulator. You will be prompted for the starting and ending address of the block by a Dialog Box. Note that the 16-byte blocking of mapping will be enforced here (if necessary) by adjusting the starting address you entered to the next lowest 16-byte boundary and the ending address to the next highest 16-byte boundary minus 1.

You will be notified of mapping constraint violations by error message.

Configure | Code Memory | Target

The *Target* command allows you to map blocks of Code Memory to the target board. You will be prompted for the starting and ending address of the block by a Dialog Box. Note that the 16-byte blocking of mapping will be enforced here (if necessary) by adjusting the starting address you entered to the next lowest 16-byte boundary and the ending address to the next highest 16-byte boundary minus 1.

You will be notified of mapping constraint violations by error message.

Configure | Code Memory | All

The *All* command calls a Pull-down Menu from which you may map all of Code Memory to either the emulator or the target board. When either is selected the Host Software will attempt to map all Code Memory to the indicated area and will enforce mapping constraint violations without any error messages being issued.

Configure | Code Memory | Load

The *Load* command allows you to load map settings from a previously saved file (via the *Configure | Code Memory | Save* or *Configure | Xdata Memory | Save* commands). You will be prompted for the filename by a Filename Dialog Box.

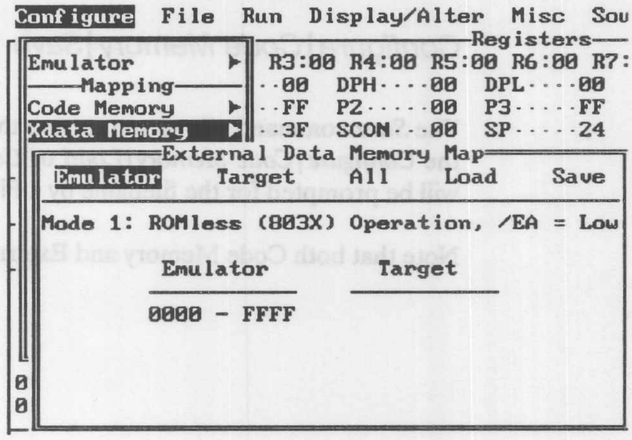
Note that both Code Memory and External Data Memory map settings are loaded from the file.

Configure|Code Memory|Save

The *Save* command allows you to save the current map settings to a file so that they may be loaded (via the *Configure|Code Memory|Load* or *Configure|Xdata Memory|Load* commands) at a later time. You will be prompted for the filename by a Filename Dialog Box.

Note that both Code Memory and External Data Memory map settings are saved to the file.

Configure | Xdata Memory



Note that the *Xdata Memory* command will be de-activated if the device in use has no ability to access External Data Memory.

The *Xdata Memory* command calls a Menu Bar Window from which you may map External Data Memory to either the emulator's External Data Memory or your target board's External Data Memory in 16-byte blocks. Changes in map settings are downloaded to the emulator as they are made.

Note that there are some mapping constraints due to emulator and device dependencies (e.g., on emulators with only 16K of External Data Memory only 16K can be mapped to the emulator). The mapping constraints are enforced. Mapping constraints (if any) are discussed in the description of each probe card in the Probe Card Reference (Chapter 4). The current mode of operation of your device is displayed in the window.

The commands are:

Configure | Xdata Memory | Emulator

The *Emulator* command allows you to map blocks of External Data Memory to the emulator. You will be prompted for the starting and ending address of the block by a Dialog Box. Note that the 16-byte blocking of mapping will be enforced here (if necessary) by adjusting the starting address you entered to the next lowest 16-byte boundary and the ending address to the next highest 16-byte boundary minus 1.

You will be notified of mapping constraint violations by error message.

Configure | Xdata Memory | Target

The *Target* command allows you to map blocks of External Data Memory to the target board. You will be prompted for the starting and ending address of the block by a Dialog Box. Note that the 16-byte blocking of mapping will be enforced here (if necessary) by adjusting the starting address you entered to the next lowest 16-byte boundary and the ending address to the next highest 16-byte boundary minus 1.

You will be notified of mapping constraint violations by error message.

Configure | Xdata Memory | All

The *All* command calls a Pull-down Menu from which you may map all of External Data Memory to either the emulator or the target board. When either is selected the Host Software will attempt to map all External Data Memory to the indicated area and will enforce mapping constraint violations without any error messages being issued.

Configure | Xdata Memory | Load

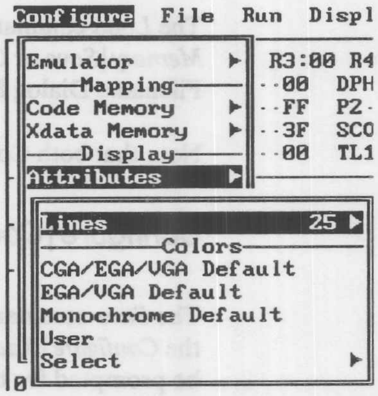
The *Load* command allows you to load map settings from a previously saved file (via the *Configure | Xdata Memory | Save* or *Configure | Code Memory | Save* commands). You will be prompted for the filename by a Filename Dialog Box.

Note that both Code Memory and External Data Memory map settings are loaded from the file.

Configure | Code Memory | Save

The *Save* command allows you to save the current map settings to a file so that they may be loaded (via the *Configure | Xdata Memory | Load* or *Configure | Code Memory | Save* commands) at a later time. You will be prompted for the filename by a Filename Dialog Box.

Note that both Code Memory and External Data Memory map settings are saved to the file.



The *Attributes* command calls a Pull-down Menu from which you may set the number of display lines and display attributes and colors. The commands are:

Configure | Attributes | Lines

The *Lines* command is used to set the number of display lines. The current number of display lines is displayed next to the *Lines* command. When the *Lines* command is selected a Pull-down Menu will list your choices. The choices available are dependent on the video display adapter installed in your Host Computer (the others will be de-activated):

MDA (Monochrome Display Adapter)	25 lines
CGA (Color Graphics Adapter)	25 lines
EGA (Enhanced Graphics Adapter)	25 lines
	43 lines
VGA (Video Graphics Array)	25 lines
	28 lines
	50 lines

Note that use of the '-v' command line option (Appendix J) will de-activate the *Lines* command.

Configure | Attributes | CGA/EGA/VGA Default

The *CGA/EGA/VGA Default* command sets up video colors and other display attributes suitable for all color monitor types. The supplied binary file $CLR1$ contains these color definitions. These definitions make use of 16 foreground colors, but only 8 background colors. If you have an EGA or VGA installed in your system, the *EGA/VGA Default* provides a full 16 background colors.

Configure | Attributes | EGA/VGA Default

The *EGA/VGA Default* command sets up video colors and other display attributes suitable for EGA and VGA color monitors. The supplied binary file $CLR2$ contains these color definitions. These definitions make full use of 16 foreground and 16 background colors. If you accidentally (or intentionally) select this command when the display adapter installed in your PC is actually a CGA or Monochrome adapter, many areas of the screen will blink! This is because the $CLR2$ file is set up assuming that the video display attribute control bit for "blink" actually means "bold (or bright) background".

Configure | Attributes | Monochrome Default

The *Monochrome Default* command sets up video display attributes suitable for monochrome adapters/monitors. The supplied binary file $CLRM$ contains these color definitions. The only "color", other than black and white, available for monochrome adapters is blue. However, "blue" does not display as blue, but rather as underlining. If you accidentally (or intentionally) select this command when the display

adapter installed in your PC is actually something other than a Monochrome adapter, the video display indeed will be painted in a rather hideous combination of black, white and blue.

Configure | Attributes | User

The *User* command sets up video colors and other display attributes to your customized settings. The binary file $COLOR (not supplied) contains these color definitions. To create your own customized, default color definitions, first invoke the *Select* command to change the video colors and display attributes to your liking.

The $COLOR file is read during iceMASTER initialization, thus establishing your own customized, default color definitions.

Configure | Attributes | Select

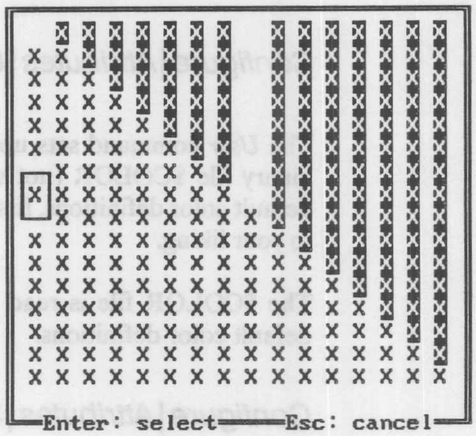
```
Use the arrow keys to move the cursor to the attribute you wish to change:
Body: Foreground (character) color
(Press 1-8, or +/-/Space to toggle;  0=blink,9=bold)      Sample Body Text:
 1-black       2-blue     3-green    4-cyan     9-bold fg      Highlight Quick
 5-red         6-magenta  7-yellow   8-white    0-                   Inact Is
Current:  white   on black
(press Enter to select colors from palette)  ─Border───  ─Body──────────────
 Area/Feature                               Cl Hl L S E  Cl Hl Qk Hk Vl In Is
 ──────────────────────────────────────────  ── ── - - -  ── ── ── ── ── ── ──
■Pull-down Menus:  Bars                                  »FB FB FB       FB FB
 Pull-down Menus:  1st level                FB FB ║ N N  FB FB FB FB FB FB FB
 Pull-down Menus:  2nd level                FB FB ║ N N  FB FB FB FB FB FB FB
 Pull-down Menus:  3rd level                FB FB ║ N N  FB FB FB FB FB FB FB
 Pull-down Menus:  4th level                FB FB ║ N N  FB FB FB FB FB FB FB
 Quick-Help line at screen bottom                        FB FB
 Function Key label lines                                FB FB          FB FB
 Pop-up:  Diagnostic Messages               FB    │ N N  FB FB
 Pop-up: Yes/No/Cancel Questions            FB    ║ N N  FB FB FB
 Pop-up:  Other Messages                    FB    │ N N  FB FB
 Dialog:  File                              FB    ║ N N  FB FB
 Dialog:  Other                             FB    ║ N N  FB FB
 Window:  Registers                         FB FB │      FB FB FB FB FB FB FB
═══════════════════════════════════════════════════════════ Esc: exit ═══
sLoad    SetBkpt ViewTrc DisAssm MacExec CurMod  SrcPath RawSrc  SymGlob1SymAlph
1Help    2ResetEm3ResetTg4Go      5GoFrom 6GoUntil7StepIns8StepLin9StepOvr0StepTo
```

The *Select* command opens a window where you may select/modify video colors and other display attributes individually. These attributes and colors will remain active until you exit the Host Software. To make these changes permanent, select the *Configure | Save | Save* command (page 7-25) to save those settings into the $COLOR file, making sure that the Save Option for Colors (*Configure | Save | Colors* on page 7-25) is ON.

Colors and attributes may be set in two ways. In both cases, you must position the cursor to the item you want to change and the results are displayed in an example box (in the top right of the window).

The first method of changing colors and attributes is to select the foreground and background colors from the color chart (in the top left of the window) by number. Foreground changes take place when you position the cursor to the left of the item, and background changes are made when you position the cursor to the right of the item, as indicated by the arrow flags.

The second method allows you to choose color combinations from a palette. After the cursor is positioned to the item you wish to change, press **Enter**. This will pop up the palette available for your monitor type. You may then use the cursor keys to position to the color combination you prefer. Press **Esc** to exit the palette without accepting your choice or **Enter** to accept your choice and exit the palette.



The column headers on the right hand side of this display are very short abbreviations for the following color controls:

Column Header	Meaning
Cl border	box border color
Hl border	box border highlight color
L border	box border line style
S border	box border, shadow under box
E border	box border, exploding box
Cl body	box body color
Hl body	box body highlight color
Qk body	box body quick-key color
Hk body	box body hot-key color
Vl body	box body value color
In body	box body inactive (inaccessible) command color
Is body	box body inactive (inaccessible) command selected (highlighted) color

You will notice that the abbreviations and the explanations refer primarily to fields that appear in Pull-down Menus, although many apply to "boxes" in general.

Some windows and menus overload the meaning of some otherwise "Not Applicable" color controls to govern a field or area which is unique to that menu or window. Those cases where the internal usage of a particular color control differs from the definitions given above are:

Function Key Label Lines:	
Cl body	Function Key numbers preceding labels
Hl body	Function Key labels
In body	Function Key label when associated command or menu is already active (invoked)

Pop-up: Yes/No/Cancel	
Hl body	currently highlighted response
Qk body	Quick Key for "Yes", "No" or "Cancel"

Window: Registers	
Hl body	highlighted "current register"

Qk body	GPRn values (left side of 1st window line)
Hk body	PSW bit values (right side of 1st window line)
Vl body	register values

Window: Source

Hl body	current PC position
Qk body	operand value history information for instruction at current PC
Hk body	module name & line number comments
Vl body	HLL source images
In body	operand value history information for instructions other than that at current PC

Window: Watch

Qk body	variable's name
Hk body	variable's value when different from value at last break
Vl body	variable's value

Window: Status

Cl body	names (labels) and all values which are not "heartbeat-updated" during emulation
Qk body	values for Real-Time Clock, Reset Count and State
Hk body	values for PC, Break Address and DPTR
Vl body	values which are "heartbeat-updated" during emulation
In body	values for Trace Memory Status, Trace Memory Read and Trace Trigger
Is body	values for Break and Repetition Counters

Menu: Help

Hl body	boldfaced text
Qk body	Hyperlink
Hk body	current ("tabbed-to") Hyperlink
Vl body	underlined text

Menu: Break-Point

Vl body	current line highlight bar

Menu: Opcode Class

Vl body	current line highlight bar

Menu: View Trace

Hl border	digital waveform probe clip display: value lines
Hk body	digital waveform probe clip display: body

Vl body HLL source images

Menu: Disassembler/Assembler

Vl body HLL source images

Menu: Code,Idata,Xdata View/Change

Hk body prompt/input line highlight Vl body, current address highlight

Menu: Performance Analyzer Setup

Vl body current line highlight bar

Menu: Performance Analyzer View

Hk body bin information (percentage) lines -- entire line except bar graph bars

Vl body HLL source images

Menu: Performance Analyzer View (Highlights)

Cl body bar graph bars

Hl body real-time Clock, Sample Count and Reset Count values (post-emulation review)

Qk body PC and Break Address values (post-emulation review)

Hk body values which are "heartbeat-updated" dynamically during emulation

Menu: Configure Windows

Vl body current values/settings

Qk body unusable window highlight

Is body unusable window selected highlight

Menu: Function Key View/Modify

Hk body highlight for current line & title in current window (Assignment or Option)

Vl body highlight for current line in non-current window.

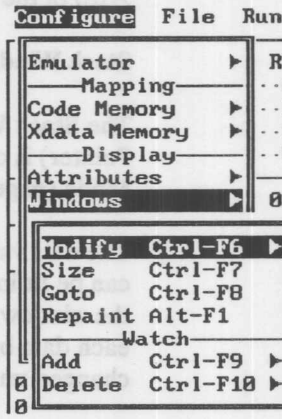
Source/Symbol Displays

Hk body accessible (can be referenced) source line numbers -- those source line numbers for which the compiler generated debugging information (used only in the Raw Source window)

Vl body current line highlight bar

Sign-On Screen

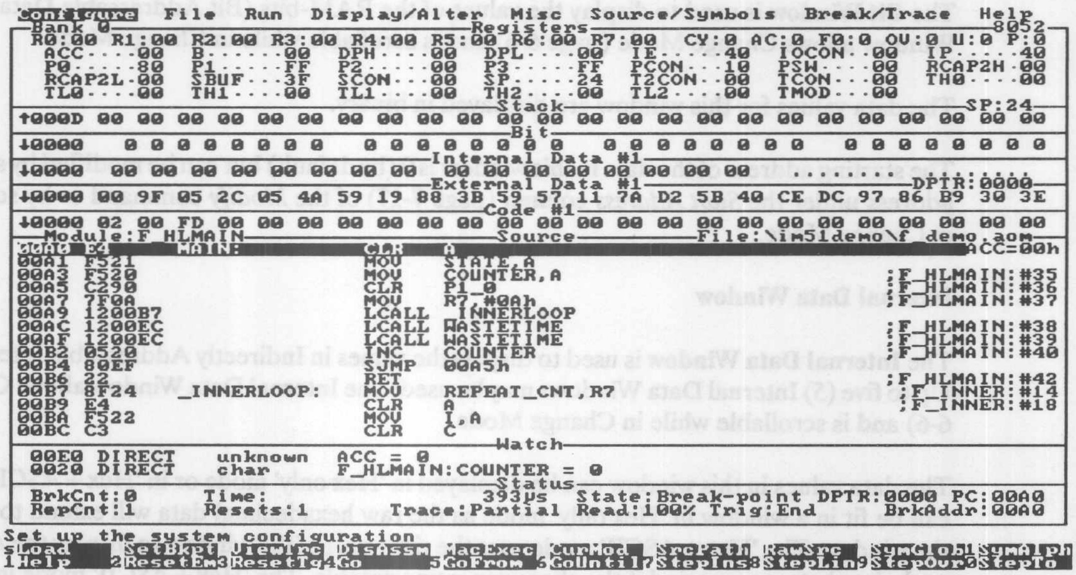
Qk body background "wallpaper"



The *Windows* command calls a Pull-down Menu from which you may modify, resize, jump into and repaint Main Screen Windows. The commands available are described after the following picture and description of each type of Main Screen Window.

The following picture shows how each type of Main Screen Window is displayed.

Main Screen Windows



Register Window

$Register WindowThe **Register Window** is used to display values for SFR's (Special Function Registers), GPR's (General Purpose Registers) and bits from the PSW (Program Status Word). The current microcontroller (defined by the $MODEL file in use) is displayed on the right side of the top border of the window. The currently selected Register Bank (0-3) is displayed on the left side of the top border of the window. The Register Window allows Change Mode (page 6-6) and is scrollable while in Change Mode.

The values of the GPR's for the current Register Bank are displayed on the left side of the first line inside the window. The bits from the PSW, other than RS0 and RS1, are displayed on the right side of the first line inside the window. The rest of the window is occupied by the SFR names and values.

The data values of the SFR's and GPR's are displayed in hexadecimal and PSW bits are displayed in binary.

Stack Window

The **Stack Window** is used to display values that have been pushed on the Stack. The value of SP (Stack Pointer) is displayed on the right side of the top border of the window. The Stack Window allows Change Mode (page 6-6) and is scrollable while in Change Mode.

The data values in this window can be displayed in 'Hex only' mode or in 'Hex + ASCII' mode. More data can be fit in a window in 'Hex only' mode as the raw hexadecimal data will extend to the right border of the window. The 'Hex + ASCII' mode uses the right part of the window to display the ASCII character for each data byte, or a period if the character is unprintable. The 'Hex only' mode is the default but can be changed under the *Options* column (page 7-18) of the *Modify* command.

The starting address of data in the window is the value of SP and cannot be changed. Note that since the Stack on the 8051 device grows upward, the value pointed to by SP will be placed at the right of the displayed line of data.

Bit Window

The **Bit Window** is used to display the values of the RAM-bits (Bit Addressable Data Memory). The Bit Window allows Change Mode (page 6-6) and is scrollable while in Change Mode.

The data values for this window are displayed in binary.

The starting address of the data in this window is 0 (by default) but can be modified by specifying a starting address under the *Start Address* column (page 7-17) of the *Modify* command or by scrolling the window in Change Mode.

Internal Data Window

The **Internal Data Window** is used to display the values in Indirectly Addressable Internal Data Memory. Up to five (5) Internal Data Windows may be used. The Internal Data Window allows Change Mode (page 6-6) and is scrollable while in Change Mode.

The data values in this window can be displayed in 'Hex only' mode or in 'Hex + ASCII' mode. More data can be fit in a window in 'Hex only' mode as the raw hexadecimal data will extend to the right border of the window. The 'Hex + ASCII' mode uses the right part of the window to display the ASCII character for each data byte, or a period if the character is unprintable. The 'Hex + ASCII' mode is the default but can be changed under the *Options* column (page 7-18) of the *Modify* command.

The starting address of the data in this window is 0 (by default) but can be modified by specifying a starting address under the *Start Address* column (page 7-17) of the *Modify* command or by scrolling the window in Change Mode.

Note that for filling, comparing and writing (to a file) blocks of Internal Data Memory the *Display/Alter|Idata* Window (page 7-38) is available.

External Data Window

The **External Data Window** is used to display the values in External Data Memory. Up to five (5) External Data Windows may be used. The External Data Window allows Change Mode (page 6-6) and is scrollable while in Change Mode.

The data values in this window can be displayed in 'Hex only' mode or in 'Hex + ASCII' mode. More data can be fit in a window in 'Hex only' mode as the raw hexadecimal data will extend to the right border of the window. The 'Hex + ASCII' mode uses the right part of the window to display the ASCII character for

each data byte, or a period if the character is unprintable. The 'Hex only' mode is the default but can be changed under the *Options* column (page 7-18) of the *Modify* command.

The starting address of the data in this window is the address pointed to by DPTR (by default) but can be modified by specifying a starting address under the *Start Address* column (page 7-17) of the *Modify* command.

Note that for filling, comparing and writing (to a file) blocks of External Data Memory the *Display/Alter|Xdata* Window (page 7-38) is available.

Code Window

The **Code Window** is used to display values in raw Code Memory. Up to five (5) Code Windows may be used. The Code Window allows Change Mode (page 6-6) and is scrollable while in Change Mode.

The data values in this window can be displayed in 'Hex only' mode or in 'Hex + ASCII' mode. More data can be fit in a window in 'Hex only' mode as the raw hexadecimal data will extend to the right border of the window. The 'Hex + ASCII' mode uses the right part of the window to display the ASCII character for each data byte, or a period if the character is unprintable. The 'Hex only' mode is the default but can be changed under the *Options* column (page 7-18) of the *Modify* command.

The starting address of the data in this window is 0 (by default) but can be modified by specifying a starting address under the *Start Address* column (page 7-17) of the *Modify* command or by scrolling the window in Change Mode.

Note that for filling, comparing and writing (to a file) blocks of Code Memory the *Display/Alter|Code* Window (page 7-38) is available.

Source Window

The **Source Window** is used to display disassembled instructions in Code Memory. Change Mode is not allowed for the Source Window. The current module (see the *Source/Symbols|Current Module* command on page 7-52) is displayed on the left side of the top border and the filename of the currently loaded program (see *File|Load* on page 7-27) is displayed on the right side of the top border.

The information in this window may be displayed in 'Code Mode' or 'Mixed Mode'. 'Code Mode' means just an assembly language disassembly of the code will be displayed. 'Mixed Mode' means that HLL source lines (if available) will be displayed along with the disassembled code. The default is 'Mixed Mode' but can be changed under the *Options* column (page 7-18) of the *Modify* command.

The starting address of the code displayed is at the PC address and cannot be changed. Note that during single step (*Run|Step* on page 7-34) or slow motion operation (*Run|Slow Motion* on page 7-34) the starting address of the screen does not change until the flow of execution moves out of the code that appears in the window, at which time a new window full of code will be displayed. This is intended to keep the Dynamically Annotated Code (see page 6-5) visible as long as possible.

Status Window

The **Status Window** is used to display information during and after emulation. The Status Window allows Change Mode (page 6-6) but not all information in the window is changeable.

The information that is changeable (in Change Mode) follows:

BrkCnt	the number of break points to pass before breaking (see *Break/Trace | Break-Count* on page 7-64)
RepCnt	the number of times to restart the program after a break (see *Run | Repetition Count* on page 7-35)
DPTR	the Data Pointer (DPH and DPL)
PC	the Program Counter (PC) address

The information that is not changeable (in Change Mode) follows:

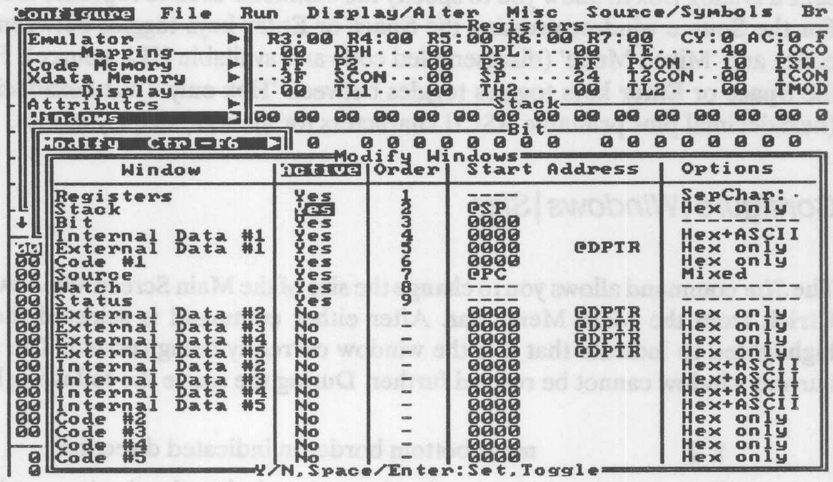
Time	execution time in microseconds
Resets	the number of resets during execution from any source (Target reset, Watchdog Timer reset, etc.)
State	during emulation the method of starting the emulation (RESET,GO) and after break the type of break (Break-point, Host-break)
BrkAddr	the break address
Trace	(Model 400 emulators only) the amount of trace data captured during the most recent emulation (Empty, Partial, Wrapped)
Read	(Model 400 emulators only) the percentage of trace data transferred to the Host Computer
Trig	(Model 400 emulators only) the type of trace trigger used for the emulation (see *Break/Trace | Trace Trigger* on page 7-64)

Watch Window

The **Watch Window** is used to keep track of program variables and/or registers during program execution. At each break the value of each item in the Watch Window will be updated. If a value has changed that value will be highlighted. Variables are added to the Watch Window using the *Configure | Window | Add* command (page 7-19) and removed from the window using the *Configure | Window | Delete* command (page 7-19).

A modified Change Mode (page 6-6) is available for the Watch Window that allows you to scroll the window but not change any values.

Configure | Windows | Modify



The *Modify* command opens a window from which you may configure various attributes of the Main Screen Windows. The window lists (column-wise) the active status, order, start address and other options for each of the Main Screen Windows. The active windows are listed in the order they appear on the Main Screen and the inactive windows are listed below them and are de-emphasized.

The ↑ and ↓ keys can be used to select which window (row) and the ← and → keys can be used to select which attribute (column) to change. The current window attribute will be marked by the highlighted column header and the highlighted attribute. Note that not all attributes of each window may be changed, as described in the description of each window. A description of each attribute follows:

Active

The *Active* column shows the active status of each window. Pressing **Y** will activate the window and pressing **N** will de-activate the window. Pressing the **Space** or **Enter** keys toggles between active and inactive. If a window is de-activated its listing will be moved below all active windows listed. When a window is made active it will be made the last listed active window.

Order

The *Order* column shows the relative position of each window. The windows are always listed in their relative position. To change the relative position of a window, press the **Space** or **Enter** keys to call a Dialog Box to enter the new position number. You may also simply press the number you want to change the position to. The window will exchange places with the window previously at the position specified.

Start Address

The *Start Address* column shows the starting address (or the method of determining the starting address) of the data displayed in each window. The Register, Status and Watch windows have no specifiable start address. The Stack window always starts at the address pointed to by SP. The Source window always contains the code at the address pointed to by PC. The External Data window can be set to always start at the address pointed to by DPTR, or can be changed to start at a specific, fixed address, as can the remaining windows.

To specify a start address (where applicable) press the **Space** or **Enter** keys to call a Dialog Box to enter an address. Pressing the **Space** or **Enter** keys while on the @DPTR symbol toggles the External Data window between the specified start address and starting at the address pointed to by DPTR.

Options

The *Options* column shows the format that the data for a window is being displayed in. There are no options for the Bit, Status and Watch windows. For the Register Window pressing the **Space** or **Enter** keys calls a Dialog Box to allow you to specify the character used to separate the register name from the value. For the Source window pressing the **Space** or **Enter** keys toggles between 'Code Mode' (disassembled code) and 'Mixed Mode' (disassembled code and available HLL source). For all other windows pressing the **Space** or **Enter** keys toggles toggles between 'Hex only' (just hexadecimal data) and 'Hex + ASCII' (hexadecimal plus printable ASCII characters for all data) display modes.

Configure | Windows | Size

The *Size* command allows you to change the size of the Main Screen Windows and is equivalent to pressing **Ctrl-R** from the Main Menu Bar. After either command is selected the top window border will be highlighted to indicate that it is the window currently being resized. The computer will beep when the current window cannot be resized further. During the resize the following keys are active:

↑ ↓	move bottom border in indicated direction
Esc	abort resize, leave the window the size it was when the command was selected
Enter	accept new window size and advance to next window
Tab	advance to next window
Shift-Tab	return to previous window

Configure | Windows | Goto

The *Goto* command activates Change Mode (page 6-6) in the top window and is equivalent to pressing **Tab** or **Shift-Tab** from the Main Menu Bar.

Once in Change Mode the following keys are active:

↑ ↓ ← →	move currently selected data item
PgUp	move up one window full of data
PgDn	move down one window full of data
Home	move to beginning of data
End	move to end of data
Esc	exit Change Mode
Enter	open a Dialog Box to enter new data
Tab	advance to next window
Shift-Tab	return to previous window

Note also that pressing any alphanumeric character while in Change Mode will open a Dialog Box to enter new data.

Configure | Windows | Repaint

The *Repaint* command repaints all Main Screen Windows and updates all data in each window. This command is useful if you want to update data values without having to go through an emulation (e.g., to check port pin values).

Configure | Windows | Add

The *Add* command is used to add a variable or register to the Watch Window. Selecting the command will call a Dialog Box for you specify the variable or register name to add.

Configure | Windows | Delete

The *Delete* command is used to remove a variable or register from the Watch Window. Selecting the command will call a Dialog Box for you specify the variable or register name to remove.

```
                         iceMASTER - 8052
          Firmware Version:  2.25;  PROM Version:  1.10
          Software Version:  3.0 Rev 11
             Emulator Base:  iceMASTER-8051 Model 400
        Model File Version:  3.014
           Emulator Memory:  64K Code, 64K External Data
      Accessible Addresses:  64K Code, 64K External Data
                     Trace:  4K
      Performance Analyzer:  High Resolution, High Bin Count
      Watchdog Timer (WDT):  Supported


                      MetaLink Corporation
                       Chandler, Arizona
               (c)Copyright MetaLink Corp. 1990,1991

                   Press any key to continue
```

The *Identification* command displays a box containing important technical information. If technical assistance is required from MetaLink, or if trouble develops with your emulator, the information presented on this screen will be very useful.

Some items, such as amount of memory installed in the emulator, firmware version, and PROM version cannot be determined unless communication between the Host Software and the emulator base has been established. If a problem develops that prevents communication from being established this information would be unavailable; therefore, we recommend that you take a moment to record all pertinent information from this screen in the space provided below. It will then be available should you need it:

Firmware Version	~~3.3 REV 6~~ 3.05
PROM Version	
Software Version	3.3 REV 6
Emulator Base Type	ICEMASTER-68HC11 MODEL 400
Model File Version	3.051
Emulator Memory Installed	64K

```
Configure File  Run  Display/Alter
                                 R
Emulator      ►  R3:00 R4:00 R5:0
     M
Code   ─────────Alert──────────────
Xdata  Diagnostics              On
----D  Bad key/command          On
Attri  Host-break               On
Windo  ──────Miscellaneous─────────
       Flicker                  On
Ident  Trace prefetch           On
Optio  Main Esc                 On
Funct  Common Window Borders    On
Save   Stack Hyperlinks         On
Resto  Prompt OS-Escape         On
       Unknown data type size  1 ►
00A7 7
```

The *Options* command calls a Pull-down Menu from which you may turn global options ON or OFF. The Alert group of options all relate to audio notification of events and are made permanent if the *Configure | Save | Save* command (page 7-25) is selected with the *Configure | Save | Alert* toggle (page 7-25) turned ON. The Miscellaneous options are made permanent if the *Configure | Save | Save* command is selected with the *Configure | Save | Misc* toggle (page 7-25) turned ON.

Configure | Options | Diagnostics

Toggle beeping ON and OFF when error messages, warning messages and important information are displayed.

Configure | Options | Bad key/command

Toggle beeping ON and OFF when an invalid key is pressed or an inactive command is selected.

Configure | Options | Host-break

Toggle beeping ON and OFF when a Host-Break (**Esc**) is issued during emulation.

Configure | Options | Flicker

Toggle ON and OFF flickering used to indicate which choice was made in a Confirmation Box.

Configure | Options | Trace prefetch

Toggle ON and OFF trace prefetch. If ON, the trace data is prefetched (read) whenever the keyboard is inactive and there is trace data to upload from the emulator to the Host Computer. Pressing a key during prefetch will interrupt the read (after a slight delay). If OFF, trace data will be fetched (read) only when you move around in the *Break/Trace | View Trace* window (page 7-65) to look at different parts of the trace buffer. There may be a slight delay as you look at different areas of trace memory.

Configure | Options | Main Esc

Toggle ON and OFF the Main Escape feature. If ON, you may exit the Host Software from the Main Menu Bar by pressing **Esc**. If OFF, you may exit the Host Software only by selecting the *File | Exit* command (page 7-32). In either case you will have to exit through a Confirmation Box. However, if you use the Hot Key assigned to *File | Exit*, **Alt-X**, no Confirmation Box will be displayed; exit is immediate.

Configure | Options | Common Window Borders

Toggle ON and OFF the Main Screen Window common borders. If ON adjacent Main Screen Windows share a common bottom and top border. If OFF all Main Screen Windows are framed with complete, separate borders.

Common Window Borders ON is the default so as to minimize the number of screen rows used for window borders and to maximize the amount of information displayed. However, when ON you will find that you probably want to configure the windows to have the same border line style and the same background color.

When OFF, this framing style uses more screen rows for window borders. However, if you choose to select different border line styles and/or background colors for each window, the results will be more pleasing visually.

Configure | Options | Stack Hyperlinks

Toggle ON and OFF the Stack Hyperlink feature. *Stack Hyperlinks* controls the behavior of the Help System when a new Help Topic is invoked via a hyperlink:

If ON, when you select a new Help Topic via a hyperlink in the current Help Topic, the host software "remembers" your position in the current Help Topic. When you exit that new Help Topic, you return to the "remembered" position in the current Help Topic. This process applies recursively.

If OFF, when you select a new Help Topic via a hyperlink in the current Help Topic, the host software does not "remember" your position in the current Help Topic. When you exit that new Help Topic, you exit the Help System.

Configure | Options | Prompt OS-Escape

Toggle ON and OFF the OS-Escape DOS prompt. If ON, during an OS-Escape, your DOS prompt will be augmented with a reminder of how to get back to iceMASTER. This reminder, added to the end of your normal DOS prompt, consists of the string:

(type 'exit' to return to iceMASTER)

If OFF, during an OS-Escape, your DOS prompt is left as is.

Configure | Options | Unknown data type size

The *Unknown data type size* command sets the number of bytes to display for the value of an unknown data type in the *Source/Symbols* (page 7-53) group of displays and in the Watch Window. The current number of bytes to display is shown next to the command.

```
Configure  File  R
 Emulator          ►
 ------Mapping------
 Code Memory       ►
 Xdata Memory      ►
 ------Display------
 Attributes        ►
 Windows           ►
                   ►
 Identification
 Options           ►
 Function-Keys     ►
                   ►
  Lines   2 ►
  Modify    ►
0
```

The *Function Keys* command calls a Pull-down Menu from which you may change the number Function Key label lines displayed and change Function Key assignments.

Configure | Function-Keys | Lines

The *Lines* command toggles the number of lines of Function Key (Hot Key) labels displayed at the bottom of the screen. This feature can be disabled (zero lines) or you may display up to four lines. If more than one line is being displayed, the character in the first column shows which key combination to use for that row of Function Keys, as follows:

A	Alt Key
C	Ctrl Key
S	Shift Key

Regardless of the number of lines of Function Key labels displayed at the bottom of the screen, all 40 Function Keys are still available for use.

Configure | Function-Keys | Modify

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
 ─────────────────────Function Key Assignment─────────────────────────────────
 Assign      Clear      Display      Save      Write      Help
  Assignments                          Options
  F1         Help      WnModfy  Configure|Windows|Modify
 ♦F2         ResetEm   WnResiz  Configure|Windows|Size
  F3         ResetTg   WnSelct  Configure|Windows|Goto
  F4         Go        WnRepnt  Configure|Windows|Repaint                      ♦
  F5         GoFrom    AddWtch  Configure|Windows|Add
  F6         GoUntil   DelWtch  Configure|Windows|Delete
  F7         StepIns   Load     File|Load
  F8         StepLin   DownLod  File|Download
  F9         StepOvr   MacExec  File|Macro|Execute
  F10        StepTo    ResetEm  Run|Reset (emulator)
  Shift-F1   Load      ResetTg  Run|Reset (target)
  Shift-F2   SetBkpt   Go       Run|Go
  Shift-F3   ViewTrc   GoFrom   Run|From
  Shift-F4   DisAssm   GoUntil  Run|Until
  Shift-F5   MacExec   GoSlow   Run|Slow Motion
 ═══════════ Ctrl-R:Resize Ctrl-V:Move  Tab,Shift-Tab:Select ══════════════════
 Assign current option to current Function Key
```

The *Modify* command calls a Menu Bar Window from which you may reassign any of the available functions to any of the 40 function keys. The window on the left side of the screen (labeled Assignments) shows the current assignments. The window on the right side of the screen (labeled Options) shows the commands available for assignment. The Tab key may be used to toggle between the two windows to scroll within each window. The commands are:

Configure | Function-Keys | Modify | Assign

The *Assign* command actually makes the assignment. Highlight the desired Hot Key, press **Tab** to change to the Option Window, then highlight the desired command and then select the *Assign* command. When you select *Assign*, the highlighted Hot Key is paired with the highlighted Option. To accept any changes for the duration of the current session it is necessary to save the assignment using the *Configure | Function-Keys | Modify | Save* command. To make the changes permanent it is necessary to save the changes to the $FKEYDEF file using the *Configure | Save | Save* command (page 7-25).

An alternate method of assignment is to highlight the desired command in the Options Window and then type the Function Key you wish it to be assigned to. For example, to assign Hot Key **Alt-F10** to the *Run | Reset | Emulator* command, highlight that command in the Options Window and then press **Alt-F10**. The assignment is now made.

Note that if you try to reassign the Hot Key for Help (by default **F1**) you will be forced through a Confirmation Box to determine if your intention is to reassign the Hot Key for Help or just to view Help information.

Configure | Function-Keys | Modify | Clear

To clear the assignment of a given Hot Key, highlight it and then select the *Clear* command.

Configure | Function-Keys | Modify | Display

The *Display* command changes how Hot Key information is displayed. The assignment procedures are the same, but the information is presented in a slightly different format. by default the short description of the command (which is displayed in the Function Key Label Lines) is shown next to the Hot Key in the Assignment Window. This command allows you to display the long description (the full command sequence) next to the Hot Key in the Assignment Window.

Configure | Function-Keys | Modify | Save

The *Save* command preserves the changes you have made for the current session. To make these changes permanent, you must select the *Configure | Save | Save* command (page 7-25) with the *Configure | Save | Function-Keys* (page 7-25) turned ON. This will write or update the $FKEYDEF file. If this file is present during Host Software initialization, Function Key assignments are read from it.

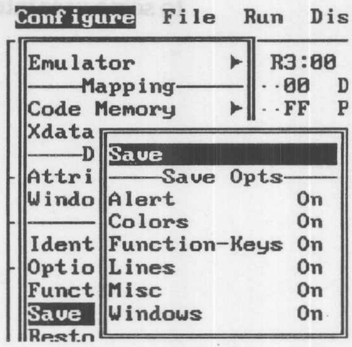
Configure | Function-Keys | Modify | Write

The *Write* command writes all current Function Key assignments to a disk file. The file is written in plain ASCII human-readable text so that you may print a copy of the file for reference.

Configure | Function-Keys | Modify | Help

The *Help* command is available to give you quick, direct access to the Help System. If, however, you press the Hot Key assigned to Help you will be forced through a Confirmation Box to determine if your intention is to reassign the Hot Key for Help or just to view Help information.

Configure | Save



The *Save* command calls a Pull-down Menu which allows you to save or update various user interface configuration options. You May toggle each of the category types ON or OFF so that only the selected changes are saved to a disk file (e.g., you may wish to save your current window and menu color choices in $COLOR, but not temporary changes to the number of screen lines).

Configure | Save | Save

This command saves information about each option whose *Save* option is ON into a specific file.

Configure | Save | Alert

When ON, *Alert* enables saving your *Configure | Options* Alert option (page 7-21) choices to the $ALERT file.

Configure | Save | Colors

When ON, *Colors* enables the saving of your color option choices to the $COLOR file. This file contains the user selected screen display attributes from the *Configure | Attributes | Select* command (page 7-9).

Configure | Save | Function-Keys

When ON, *Function-Keys* enables the saving of your Hot Key assignments to the $FKEYDEF file. Function Key assignments can be changed using the *Configure | Function-Keys | Modify* menu (page 7-23).

Configure | Save | Lines

When ON, *Lines* enables the saving of the number of screen lines (video mode) to the $LINES file. The number of screen lines can be changed (depending on video adaptor installed) using the *Configure | Attributes | Lines* Pull-down Menu (page 7-8).

Configure | Save | Misc

When ON, *Misc* enables the saving of your (*Configure | Options*) Miscellaneous option (page 7-21) choices to the $MISC file.

Configure | Save | Windows

When ON, *Windows* enables the saving of window size and position information to the $WINDOW file.

Configure | Restore

The *Restore* command restores configuration choices previously saved using the *Configure | Save | Save* command (page 7-25), without exiting and reentering the Host Software. If you make temporary changes to some user interface configuration option, you may restore your permanent choices with this command.

File

The *File* command calls a Pull-down Menu from which you may load a program, save or restore the emulation environment, upload and download code and data to (or from) your target system, create and execute macros to automatically handle frequently repeated tasks, temporarily exit to DOS from the Host Software and Exit the Host Software. The commands available are:

File | Load

The *Load* command allows you to load a program into the emulation environment. The special Filename Dialog Box is used to prompt you for the filename (see the Software Guide). Remember that a directory listing is available using the ? (terse) and ! (verbose) commands.

After a valid file name has been entered, a Confirmation Box will ask if you wish to

Merge into the current application environment?

If you answer **Y** then all breakpoint, map, and trace (Model 400 emulators only) settings will be kept, and the code and symbols in your program will be merged with any existing program. Code at conflicting addresses from the new file will overwrite the same code in the old file. All symbolic and HLL information from the old file will remain and any such information in the new file will be added.

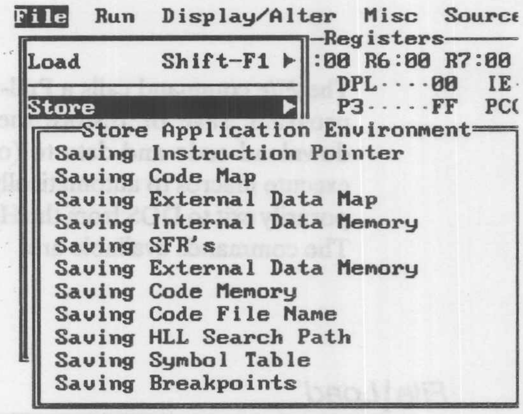
If you answer **N** the previous environment including break, trace (Model 400 emulators only), and code memory will be cleared or restored to default settings before the new program is loaded. Pressing **Esc** cancels the *Load* operation.

Please note that this command only loads your program into the Host Computer and the emulator's code memory. If your target system is set up in a Von Neuman configuration (#PSEN and #WR 'anded' together), and you wish the code to be placed in target RAM, you should use the *File | Download* command (page 7-29).

While your program is being loaded, you will see address ranges displayed on the right of the Quick Help Line and source line and module information on the left, if found. Assembly language programs will never have line number information.

If your HLL generated program doesn't display source line or module information, you should check to be sure that the source files can be found in the HLL search path (see *Source/Symbols | Source Path* on page 7-52). If the HLL search path is set properly refer to Recommended Compilation Options (Appendix H) to verify the compilation and linker options.

File | Store



The *Store* command allows you to store the emulation environment to a file for later recall by the *File | Restore* command (below). You will be prompted for the filename by a Filename Dialog Box.

While the *Store* operation is taking place a Status Box will display each item as it is being saved. The emulation environment is defined by:

Current PC (program counter)

Code map

External Data map

Internal Data memory

SFR values

External Data memory

Code memory

Code file name

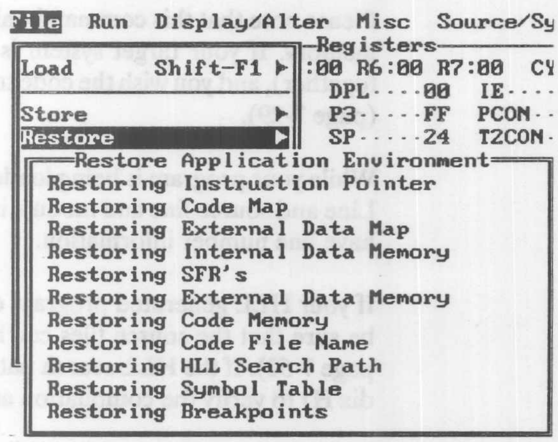
HLL search path

Symbol table

Breakpoint settings

If the device being used can access external data memory, a Dialog Box will ask how many K bytes to save (from 0-64 valid).

File | Restore



The *Restore* command is used to restore the emulation environment from a file saved using the *File | Store* command (above). You will be prompted for the filename by a Filename Dialog Box.

File | Upload

The *Upload* command allows you to upload code memory from the target system into the emulator's code memory space, where it can be modified. When you select this command you will be asked for the range of addresses you wish to upload. After responding a Confirmation Box will ask:

Merge into the current application environment?

A Y answer will cause existing code within the specified upload range to be overwritten by the uploaded code. Current breakpoints and labels will be retained as will trace ON/OFF (Model 400 emulators only) points and HLL line number references.

A N answer will clear all of the emulation environment settings currently in effect along with code memory and any symbolic or HLL line number references.

File | Download

The *Download* command is used to download code or external data from a file on the host computer to the target system. You will be prompted for the filename by a Filename Dialog Box, and after entering a valid filename a Confirmation Box will ask:

Are you downloading code memory?

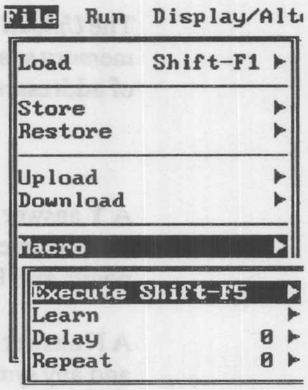
A Y answer means download to the target board's code memory and a Confirmation Box will ask:

Merge into the current application environment?

A Y answer merges information from the file into the current application environment and N clears the application environment before loading information from the file. For more information on the merge prompt see the *File | Load* command (page 7-27). Note that your target must be designed as a Von Neuman architecture (#PSEN and #WR 'anded' together) to actually download any code to the target system memory.

A N answer to the download prompt means download to the target board's external data memory. The information from the file will be downloaded to the target board. If the file contains symbolic information, all symbolic information will be ignored.

On the surface, it may not seem necessary to worry about the emulation environment if you are downloading code to the outside world, but the Host Computer must know if it is to clear symbolic information, breaks, or Trace ON/OFF points (Model 400 emulators only) from the emulator at load time.



The *Macro* command calls a Pull-down Menu from which you may create and execute macro command files. A macro command file contains a series of recorded keystrokes which, when executed, perform a macro function. A macro command file may be used to automate long or repetitive tasks.

Macro Environment

When the emulator is in macro learn mode, all keystrokes typed are saved to a disk file in order to be reproduced (executed) at a later time. Not only are the keystrokes saved, but the Function Key assignments at macro creation time are saved as well. Because of this, Function Key keystrokes will perform the commands assigned to them at the time the macro file was created. If the Function Keys have been redefined, the user may notice that the macro file Function Keys may not perform the same tasks as the currently defined ones. At macro completion, the Function Keys will be returned to their redefined values.

Note that for macro files that may be used over a long period of time it would be wise to avoid using cursor keys to select commands and to always use the Quick Key (the highlighted letter of the command). This would be useful in the situation where a Host Software update occurred where several commands have been added. Execution of a macro file using cursor keys may get 'out-of-synch', meaning because of the extra commands, commands other than those intended may be selected.

Parameter Substitution

If the same task needs to be performed repeatedly, but values may change between runs, the same macro command file can be executed using parameters. In general, parameters can be used anywhere you are prompted for information (filenames, numeric values, addresses, etc.). When creating a macro command file, a **%P** (see the *File | Macro | Learn* command below) is keyed in wherever a parameter value is to be used. When a macro command file is executing, it substitutes a parameter value whenever the **%P** keystroke combination is encountered. A parameter list may be supplied when invoking the macro command file (see the *File | Macro | Execute* command below). During macro execution, if no parameters are supplied and **%P** is encountered, execution stops until a parameter is provided.

File | Macro | Execute

The *Execute* command begins execution of the commands in a macro command file created using the *Learn* command. After entering the name of a valid macro command file you will be prompted for any parameters to pass to the macro command file.

All parameters should be listed in the order that they are used in the macro command file. Each parameter will be prompted for separately. Pressing **Enter** in response to a parameter prompt (without entering any data) will start execution of the macro command file.

Commands are executed from the macro command file until the end of the file is reached or **Ctrl-C** is pressed. Control then returns to you (manual entry).

Note that you may use the '-i' command line option (Appendix J)to start automatic execution of a macro command file when you invoke the Host Software.

File | Macro | Learn

The *Learn* command is used to create a macro command file. After entering a valid file name control is returned to the Main Menu Bar and the system enters Learn Mode. In Learn Mode every keystroke entered is stored in the macro command file until **%E** (see Learn Mode Commands below) is entered. All Host Software commands are accessible in Learn Mode except for the *File | Macro* command.

Learn Mode Commands

%E	End macro learn mode
%%	Insert a % sign in the macro file as normal text.
%P	Parameter substitution. Everything typed up to the next **Enter** will be passed to the program and not included in the macro file. When the **Enter** key is pressed, Learn Mode resumes. During execution a parameter will be pulled from the parameter list or, if no parameters are present, the emulator will wait for the user to supply one.
%D	Enter a delay. Everything typed up to the first non-digit character is a real-time delay (in milliseconds). This delay is not the same as the global delay (*File | Macro | Delay*). This delay is used only between the keystrokes in which it was entered in the macro file. This local delay is added to the global delay, if one was specified.
%N	Execute a new macro. Everything typed up to the **Enter** is the name of a new macro file. If entered during Learn Mode, the macro command file is closed and Learn Mode ends. If entered during execution of a macro, the current macro command file is closed, the new macro command file is opened, and execution continues from the new file.

NOTE: No keystrokes will be echoed to the display when entering Learn Mode command information. Any other % key combinations will be ignored.

File | Macro | Delay

The *Delay* command is used to set a global delay time (between each keystroke) during macro command execution. Valid delay times range from 0 to 15000 milliseconds (15 seconds). The default value for the global delay is 0.

File | Macro | Repeat

The *Repeat* command is used to set the number of times that the execution of a macro command file is to be repeated. Valid repeat counts range from 0 to 60000. The default repeat count is 0.

File | OS-Escape

The *OS-Escape* command temporarily exits the Host Software to DOS while keeping the Host Software system intact. An error message will be reported if there is not enough memory available to escape to DOS. This may happen if a program is loaded with a large amount of symbolic information or if many TSR's are loaded on the Host Computer. After completing your DOS tasks, type **exit** to return to the Host Software.

By default the *OS-Escape* command adds the prompt

(Type exit to return to iceMASTER)

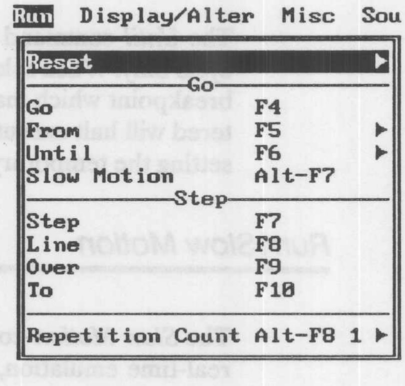
to your DOS prompt. Note that if you do not want to modify your DOS prompt, this feature may be disabled using the *Configure | Options | Prompt OS-Escape* command (page 7-22).

File | Exit

The *Exit* command allows you to exit the Host Software. Note that it is also possible to exit the Host Software from the Main Menu Bar by pressing **Esc** if the *Configure | Options | Main Esc* option is ON (page 7-21). In either case you will be forced through a Confirmation Box to confirm your intention to exit the Host Software.

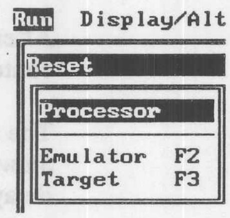
Alternatively, if you use the Hot Key assigned to *File | Exit* (**Alt-X**), exit from the Host Software is immediate, without the Confirmation Box.

Run



The *Run* command calls a Pull-down Menu from which you can select commands to begin or resume emulation (execution of your program in the emulator). The commands are:

Run|Reset



The *Reset* command calls a Pull-down Menu from which you may select one of the three types of **Resets** available. They are:

Run|Reset|Processor

The *Processor* command resets the microcontroller in the probe card to its **Reset** state. Unlike the *Emulator* and *Target* commands (below), emulation does not actually begin.

Run|Reset|Emulator

The *Emulator* command starts emulation from a device **Reset** condition. The **Reset** signal comes from the emulator and occurs automatically. There is no output to synchronize the target system, as the reset pin to the target is input only. Some devices have a **Reset** out signal on a separate pin. This signal, if present in the microcontroller itself, will be available.

Run|Reset|Target

The *Target* command also starts the microcontroller from a **Reset** condition, but the source of the **Reset** signal is the target system or the **Reset** button on the emulator. Emulation will not begin until the **Reset** signal is supplied.

Run|Go

The *Go* command begins (or restarts) emulation starting from the current program counter (PC) address. If the emulation session is started with a *Go* command from an unknown state (i.e., without having been **Reset** first, unpredictable behavior may result.

Run|From

The *From* command allows you to begin (or restart) emulation at a program counter (PC) address which you will be prompted to supply. Aside from changing the program counter (PC) address, this command works just like the *Go* command (above).

Run | Until

The *Until* command allows you to set a temporary, simple breakpoint which is valid for one emulation cycle only. When selected, a Dialog Box will prompt you for the temporary breakpoint address. No other breakpoint which may be set is affected. When the emulation begins, the first active breakpoint encountered will halt execution in the usual manner and this temporary breakpoint will be cleared. Aside from setting the temporary breakpoint address, this command works just like the *Go* command (above).

Run | Slow Motion

The *Slow Motion* command starts an automatically repeated cycle of *Run | Step* commands. This is not real-time emulation, since the system executes only one instruction, then breaks and updates the Main Screen Windows, then continues. A *Slow Motion* emulation is stopped by pressing **Esc**.

After each instruction, the right side of the Source Window is used to display memory and register contents and program flow information. This is referred to as **Dynamically Annotated Code** (page 6-5).

The speed of operation of the *Slow Motion* emulation is under your control as well. While operating in *Slow Motion*, use the + and - keys (on the keypad) to speed up or slow down the speed of execution. The delay between each *Step* can be varied from 100 to 2,000 (.1 to 2 seconds) and the default is 500 (.5 seconds). The current delay value is shown in the Status Window as the state. When changing speed, note that only one change increment is allowed per *Step*.

The *Slow Motion* command updates the entire display after each instruction. If you do not need real-time speed, this mode used in conjunction with the Watch Window, allows you to identify problems with unexpected data values or register value changes. All other break conditions are de-activated.

Run | Step

The *Step* command is a **Single Step** and is used to execute one assembly (machine language) instruction and then break emulation. The instruction that will be executed is at the PC address.

An additional note on **Single Step** operation: This is not, strictly speaking, real-time emulation. Timer-related programs may not operate correctly when single stepped. This may also include edge-triggered interrupts, serial port functions, and timer operation. See Chapter 8, Run-Time Considerations for additional information.

Run | Line

The *Line* command is used to **Single Step** by source line number. Emulation begins at the current PC and breaks when execution reaches the next source line of **any** module.

Run | Over

The *Over* command is used to **Single Step** by source line number. Emulation begins at the current PC and breaks when execution reaches the next source line of the **current** (same) module. This has the effect of "stepping over" calls to procedures or function in other modules.

Run | To

The *To* command is used to **Single Step** by procedure or function entry points. Emulation begins at the current PC and breaks when execution reaches the next function, procedure or global code label. Note that global (public) code labels are included only because not all language processors (compilers) mark procedures and functions properly in the load file (see Appendix G, HLL Support Of Third Party Software).

Run | Repetition Count

The *Repetition Count* command is used to specify how many emulation cycles will occur after the next run-type (*Reset*, *Go* or *Step*) command is issued before emulation stops completely. After a breakpoint is encountered, emulation will halt for a moment, the data in all Main Screen Windows will be updated and if the repetition counter is non-zero, it is decremented and execution will continue. If the repetition counter is zero at this point, emulation will not be continued. The repetition counter may be set to any value between 0 to 65535, inclusive.

Host-break

While code is executing in the emulator, regardless of how it was started (*Reset*, *Go* or *Step*), the **Esc** key can be used to force the emulator into break Condition. The break occurs at the nearest **opcode fetch**. The message:

Host-break

is displayed for the state in the Status Window. The Trace Trigger (see *Break/Trace | Trace Trigger* on page 7-64) is forced to **End**. Pressing the **Break** push-button on the emulator chassis during emulation is equivalent to pressing **Esc** during emulation.

```
Display/Alter  Misc
┌──────────────────────┐
│Asm/Dasm Shift-F4 ►   │
│       Memory         │
│Code     Alt-F2   ►   │
│Idata    Alt-F3   ►   │
│Xdata    Alt-F4   ►   │
│                      │
│Var/Reg  Alt-F5   ►   │
│                      │
│RAM-Bits Alt-F6   ►   │
└──────────────────────┘
```

The *Display/Alter* command calls a Pull-down Menu from which you may assemble new code into memory, disassemble existing code memory, view/change variable and register contents, view/change RAM-Bits, view/change raw Code, External Data and Internal Data memory. The commands are:

Display/Alter|Asm/Dasm

The *Asm/Dasm* command calls a Menu Bar Window from which you may assemble instructions into code memory and disassemble existing code memory. By default the Disassemble Mode is active on the first entry to the window but on subsequent entrances to the window the active mode will be whichever mode (Disassemble or Assemble) was active on the most recent exit from the window. The label in the center of the top border of the window shows the current active mode.

The right side of the top border line (under the Menu Bar) shows the filename of any loaded code file. The left side of the bottom border line shows the Display Mode. The center of the bottom border line shows current key information. The right side of the bottom border line shows the Label-synch mode. The commands are:

Display/Alter|Asm/Dasm|Disassemble

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
┌─────────────────────────────Disassembler──────────────────────────────────┐
│ Disassemble     Assemble     Mode     Label-synch     Write               │
├─Module:F_HLMAIN──────────────────────────File:\im51demo\f_demo.aom────────┤
│Addr Code   Label             Instruction       Start: 00A0                │
│──── ────── ──────────────── ─────────────────────────────────────────────│
│00A0 E4     MAIN:             CLR    A                          ;F_HLMAIN:#33│
│00A1 F521                     MOV    STATE,A                               │
│00A3 F520                     MOV    COUNTER,A                  ;F_HLMAIN:#35│
│00A5 C290                     CLR    P1_0                       ;F_HLMAIN:#36│
│00A7 7F0A                     MOV    R7,#0Ah                    ;F_HLMAIN:#37│
│00A9 1200B7                   LCALL  _INNERLOOP                            │
│00AC 1200EC                   LCALL  WASTETIME                  ;F_HLMAIN:#38│
│00AF 1200EC                   LCALL  WASTETIME                  ;F_HLMAIN:#39│
│00B2 0520                     INC    COUNTER                    ;F_HLMAIN:#40│
│00B4 80EF                     SJMP   00A5h                                 │
│00B6 22                       RET                               ;F_HLMAIN:#42│
│00B7 8F24   _INNERLOOP:       MOV    REPEAT_CNT,R7               ;F_INNER:#14│
│00B9 E4                       CLR    A                           ;F_INNER:#18│
│00BA F522                     MOV    I,A                                   │
│00BC C3                       CLR    C                                     │
└──Mode=Code──────────── PgDn:Next Screenful ─────────────Label-Synch=Off───┘
Disassembly Mode
sLoad    SetBkpt  ViewTrc  DisAssm  MacExec  CurMod   SrcPath  RawSrc   SymGlob1SymAlph
1Help   2ResetEm3ResetTg4Go       5GoFrom  6GoUntil7StepIns8StepLin9StepOvr0StepTo
```

The *Disassemble* command turns on Disassemble Mode. In this mode you can quickly view a page of disassembled code from anywhere in code memory by typing an absolute code address or a symbolic name (line number or label), or can scroll through code memory (disassembling a page at a time) using the **PgDn** key. When Disassemble Mode is made active a window full of code will be displayed.

The disassembled code will be interspersed with HLL source images (if available) if the 'Mixed' display Mode is active.

Note that if an absolute address is entered and the Label-synch Mode is OFF, the address may or may not correspond to an instruction boundary and may result in an 'out-of-synch' disassembly.

Display/Alter | Asm/Dasm | Assemble

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
                              Assembler
 Disassemble     Assemble      Mode     Label-synch     Write
 Module:F_HLMAIN                              File:\im51demo\f_demo.aom
Addr Code   Label            Instruction           Start: 0b7
                                         New Instruction
00B7 8F24   _INNERLOOP:      MOV    R  MOV REPEAT_CNT,5          F_INNER:#14
00B9 E4                      CLR    A                             F_INNER:#18
00BA F522                    MOV    I,A
00BC C3                      CLR    C
00BD E524                    MOV    A,REPEAT_CNT
00BF 6480                    XRL    A,#80h
00C1 F8                      MOV    R0,A
00C2 E522                    MOV    A,I
00C4 6480                    XRL    A,#80h
00C6 98                      SUBB   A,R0
00C7 5022                    JNC    00EBh
00C9 1200EC                  LCALL  WASTETIME                    ;F_INNER:#19
00CC 309004                  JNB    P1_0,00D3h                   ;F_INNER:#20
00CF C290                    CLR    P1_0                         ;F_INNER:#21
00D1 8002                    SJMP   00D5h
 Mode=Code                  ↓↑:Inc/Dec Start           Label-Synch=Off
Assembly Mode
sLoad    SetBkpt  ViewTrc  DisAssm  MacExec  CurMod   SrcPath  RawSrc   SymGlob 1SymAlph
1Help   2ResetEm 3ResetTg 4Go      5GoFrom  6GoUntil 7StepIns 8StepLin 9StepOvr 0StepTo
```

The *Assemble* command turns on Assemble Mode. In this mode you may enter new instructions into code memory a single line at a time. The assembler may also be used to add a label at a specified PC address. The PC address at which to add the instruction may be specified by typing an absolute code address, a symbolic name (line number or label) or by using the ↓ and ↑ keys to increment and decrement the address. When Assemble Mode is made active the current disassembly data is left in the window for reference.

When entering an instruction, symbolic information may be used. This includes the ability to define a new label at the current address and using symbolic names in the operand part of the instruction. When a new label is entered, it is inserted into the internal symbol table as a global (public) symbol.

Note that if the assembled instruction does not contain the same number of bytes as the original instruction at the current address you will be notified and asked to verify the operation through a Confirmation Box.

Display/Alter | Asm/Dasm | Mode

The *Mode* command is used to toggle the display mode between 'Code' Mode and 'Mixed' Mode. The 'Code' display Mode means that just assembly language instructions are displayed. The 'Mixed' display Mode (available only if HLL source is available) means that HLL source images are interspersed with the assembly code.

Display/Alter | Asm/Dasm | Label-synch

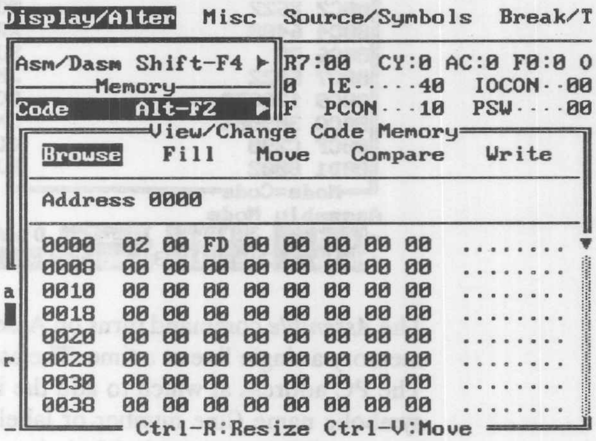
The *Label-synch* command is used to toggle label synchronization ON and OFF. When Label-synch is ON, the software uses existing code labels to verify that disassembly addresses are on instruction boundaries. If they are not the software will adjust the address so the disassembly begins on an instruction boundary. If the Label-synch mode is OFF, no such checks are made and the disassembly will start at the specified address.

Note that if Label-synch Mode is OFF, the specified address may or may not correspond to an instruction boundary and may result in an 'out-of-synch' disassembly.

Display/Alter|Asm/Dasm|Write

The *Write* command is used to write the disassembled contents of code memory to a disk file. The format of the information written to that file will be just as you see it on the screen (i.e., in human-readable form), in the current display mode.

Display/Alter|Code
Display/Alter|Idata
Display/Alter|Xdata



The *Code*, *Idata* and *Xdata* commands call Menu Bar Windows which are used to perform nearly identical functions, only on three separate memory spaces (Code, Indirectly Addressable Internal Data and External Data). Where the functions differ, the descriptions will be separate. In general, these commands are intended to manipulate raw memory in various ways.

Note that the *Xdata* command will be de-activated for devices that have no ability to access External Data Memory.

The commands available are:

Display/Alter|Code|Browse
Display/Alter|Idata|Browse
Display/Alter|Xdata|Browse

The *Browse* command is used to enter Browse Mode. In Browse Mode you can scroll through memory by using any of the keypad cursor keys or can type an address (absolute or symbolic) to quickly view an area in memory. Note that Browse Mode may also be entered automatically from the Menu Bar by pressing any of the keypad cursor keys (except the ← and → keys).

Once in Browse Mode, you may switch between Browse Mode and Change Mode (page 6-6) by pressing the **Tab** or **Shift-Tab** keys. In Change Mode you may scroll through memory in the same manner as Browse Mode but may change the value of a memory address. When positioned at the desired address, pressing the **Enter** key or any alphanumeric key will call a Dialog Box to enter a new value to store at that address.

Display/Alter|Code|Fill
Display/Alter|Idata|Fill
Display/Alter|Xdata|Fill

The *Fill* command lets you quickly fill a block a memory with some value (or values). Selecting the *Fill* command calls a Dialog Box to prompt for the address range to fill and the pattern to fill that range with.

The fill pattern may contain one or many values up to a total typed length of about 80 characters. If multiple values are needed, they must be separated by commas or spaces. Numeric values are by default hexadecimal and ASCII literals must be quoted (e.g., use 'A' to put the ASCII value of A in memory). The allowed characters in a fill pattern are listed in Appendix D, Character Sets.

Display/Alter | Code | Move
Display/Alter | Idata | Move
Display/Alter | Xdata | Move

The *Move* command is used to move (copy) a block of memory to another location in memory. Selecting the *Move* command will call a Dialog Box to prompt for the address range for the source block and for the starting address of the target block.

Display/Alter | Code | Compare
Display/Alter | Idata | Compare
Display/Alter | Xdata | Compare

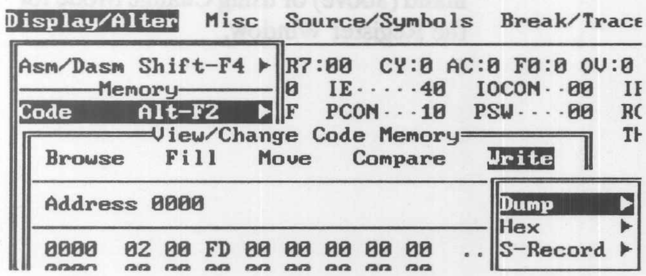
The *Compare* command is used to compare two blocks of memory. Selecting the *Compare* command will call a Dialog Box to prompt for the address range of one block and the starting address of the second block. If the blocks are the same a message will inform you of that. If the blocks are not the same, the addresses and values of the mismatches will be displayed, along with a count of the total number of mismatches found.

Display/Alter | Idata | Write
Display/Alter | Xdata | Write

The *Write* command (for Internal and External Data Memory) is used to write a block of memory to a disk file. The block will be written in the same format that the data is displayed on screen. When you select the *Write* command, a Dialog Box will prompt you for the address range of the block to write and for the name of the file to write the data to.

Display/Alter | Code | Write

The *Write* command (for Code Memory) calls a Pull-down Menu from which you may write code memory to a disk file in a Dump format, a reloadable Hex format and a reloadable S-Record format.



The commands are:

Display/Alter | Code | Write | Dump

The *Dump* command is used to to write a block of code memory to a disk file. The block will be written in the same format that the data is displayed on screen. When you select the *Dump* command, a Dialog Box will prompt you for the address range of the block to write and for the name of the file to write the data to.

Display/Alter | Code | Write | Hex

The *Hex* command is used to write code memory to a reloadable disk file in Intel Standard Hex format (see Appendix F, File Formats). When you select the *Hex* com-

Display/Alter | Code | Write | S-Record

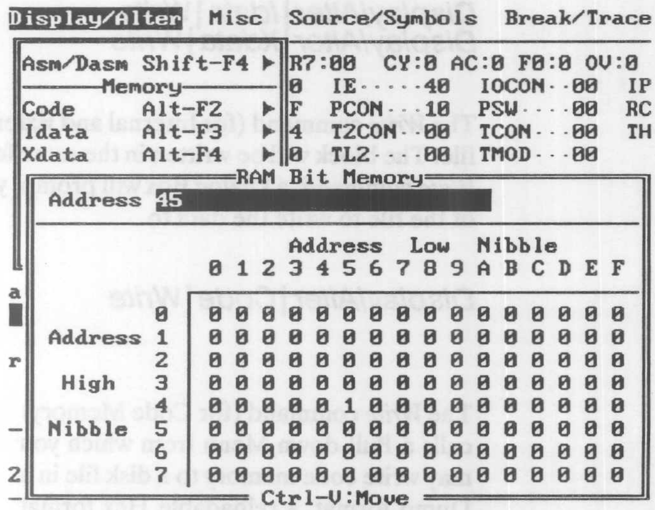
The *S-Record* command is used to write code memory to a reloadable disk file in Motorola S-Record format (see Appendix F, File Formats). When you select the *S-Record* command, a Dialog Box will prompt you for the address range of the block to write and for the name of the file to write the data to.

Display/Alter | Var/Reg

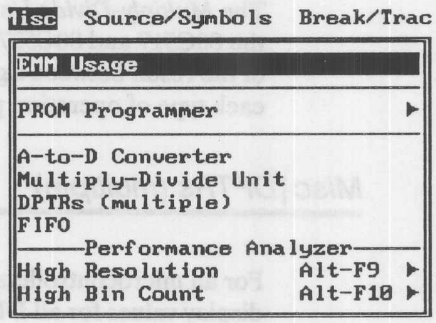
The *Var/Reg* command can be used to change the value of any program variable (including bits) or register (including the PC). When you select this command you will be prompted for the name of the variable or register to change and for its new value. The current value will also be displayed.

Display/Alter | Ram-Bits

The *RAM-Bits* command is used to view or change bits in the 128 directly-addressable bit area of Internal Data memory. A bit's value may be toggled by typing the bit address (absolute or symbolic) or by using the cursor keys to move to the desired bit and then pressing the **Enter** key. Note that SFR bits can be changed using the *Var/Reg* command (above) or using Change Mode for the Register Window.



The *Misc* command calls a Pull-down Menu from which you have access to:



1) unique features of the microcontroller
2) special options/features of the iceMASTER emulator
3) miscellaneous other things

Misc|EMM Usage

If your Host Computer has LIM-compatible expanded memory, the iceMASTER Host Software can make limited use of it. To enable access to expanded memory, you must have the appropriate "Device =" statement in your CONFIG.SYS file. See the documentation provided by the manufacturer of your expanded memory for specific details.

The *EMM Usage* command displays the following information:

System	The total amount of expanded memory present in your Host Computer.
Other	The amount of expanded memory being used by programs other than the iceMASTER Host Software.
Usable	The amount of expanded memory which can be used by the Host Software.
In Use	The amount of expanded memory actually being used by the Host Software. The **In Use** value will never exceed the value for **Usable.**

Misc|PROM Programmer

The *PROM Programmer* command is used to operate a special piece of hardware with which you may program 83C751 or 83C752 PROM's and EPROM's. For those who buy the special hardware, a write-up describing the use of the *PROM Programmer* command is available.

Misc|A to D Converter

The *A to D Converter* command is used to read all analog to digital converter channels and display each channel's digital value in hexadecimal.

Note: The reading of the analog to digital converter requires that certain registers be set in the proper state to activate this function. Executing the *A to D Converter* command will modify those registers. Refer to the data specification for the microcontroller in use to determine which registers may be affected.

Misc | Multiply-Divide Unit

The *Multiply-Divide Unit* command is used to display the formatted result of an arithmetic operation of the 80C517 and 80C537 Multiply-Divide Unit (MDU). Pressing the **Enter** key toggles the interpretation of the result between signed decimal and unsigned decimal format. The result is displayed in a format for each type of operation possible for the MDU.

Misc | DPTRs (multiple)

For all microcontrollers that have more than one data pointer, the *DPTRs (multiple)* command is used to display values for all DPTRs (data pointers) in hexadecimal. The values for the selector register are also displayed in hexadecimal.

Misc | FIFO

The *FIFO* command is used to read and display values from the FIFO feature of your microcontroller (if present). The values are displayed in hexadecimal. Note that once the FIFO is read the data in the FIFO is lost as the read is destructive and the data cannot be restored.

Misc | High Resolution (Model 400 emulators only)
Misc | High Bin Count (Model 400 emulators only)

Performance Analyzer Overview

The performance analyzer allows you to monitor the amount of time spent executing specified parts of the program loaded in code memory. There are several reasons for using a performance analyzer:

1) To identify and optimize program "hot spots".
2) To verify there is no "dead" (unexecuted) code.
3) To verify that execution is always within program bounds.
4) To verify execution of certain code in response to some internal or external condition/stimulus.

There are two types of performance analysis available, a **High Resolution Performance Analysis** and a **High Bin Count Performance Analysis**. The High Resolution Performance Analysis has 15 bins available but is very accurate (the PC is sampled approximately every 5.4 microseconds). The High Bin Count Performance Analysis has 999 bins available but the PC is sampled mush less often.

Each portion of the program to be monitored is called a bin. Usually, each bin will consist of a single code memory address range (starting address through ending address, inclusive), although a bin may be created with several code memory address ranges which are not contiguous. PC sample counts and percentages are collected on a per-bin basis.

Bin Layout

A bin contains a Type, a Number, a Capture Range and a Description, as follows:

Bin Types **Miss Bin**

The Miss Bin is automatically added to a setup if the defined Capture Ranges do not entirely cover code memory (i.e., if there are gaps in the address ranges to be monitored). This may occur after a bin has been Edited or Added, after a Quick-setup has been defined, or after a setup has been Loaded.

The Miss Bin is assigned to Bin Number 15 for the High Resolution Performance Analyzer setup and to Bin Number 999 for the High Bin Count Performance Analyzer setup. You cannot modify the Miss Bin.

Break Bin

The Break Bin is available only for the High Resolution Performance Analyzer setup since the normal break-point mechanism is inoperable when performing a High Resolution Performance Analyzer setup. The Break Bin is always Bin Number 16. You can specify any number of break-points (or ranges) using the Break Bin. This bin may be created by Adding a bin and setting the bin number to 16, or by Editing an existing bin and changing its bin number to 16.

During High Resolution Performance Analyzer emulation if the program executes at any code location monitored by the Break Bin, emulation will break (stop) as with 'normal' break-points.

User Bin

The User Bin is any bin that you have Added or Edited, other than the Break Bin. If a bin created by a Quick-setup is later Edited its Bin Type will be changed to User.

Neql Bin

The Neql Bin is a bin that was created using the *Quick-setup|N-Equal* command (page 7-46).

Mod Bin

The Mod Bin is a bin that was created using the *Quick-setup|Module* command (page 7-46).

Proc Bin

The Proc Bin is a bin that was created using the *Quick-setup|Procedure* command (page 7-46).

Lnum Bin

The Lnum Bin is a bin that was created using the *Quick-setup|Line Numbers* command (page 7-47).

Bin Number

The Bin Number can be any number greater than 0 and less than or equal to the maximum number of bins. The maximum number of bins allowed is normally 15 for the High Resolution Performance Analyzer setup and 999 for the High Bin Count Performance Analyzer setup. If the current setup requires a Miss Bin, the maximum number of bins allowed is 14 for a High Resolution Performance Analyzer setup and 998 for a High Bin Count Performance Analyzer setup as the miss bin number is 15 for the High Resolution Performance Analyzer setup and 999 for the High Bin Count Performance Analyzer setup.

In the case of a High Resolution Performance Analyzer setup, setting the bin number to 16 is legal and causes a Break Bin to be created.

Bin Capture Range

The Capture Range is the range of code addresses monitored by a bin (for which sample counts are captured). All Capture Ranges (for all bins) will always be within the bounds of the Capture Span. In addition, no Capture Range will ever overlap any other Capture Ranges. Note that there may be several distinct Capture Ranges for the same bin number depending on how the setup was created.

When Editing a bins Capture Range several checks are made to determine if the range is valid. The ending address of the range must be equal to or greater than the starting address range and the range may not overlap any other range, except for any ranges in Miss Bins.

Bin Description The Bin Description field allows you to enter a descriptive tag for a bin. Up to 30 characters can be displayed. The Bin Description field is also used by the Quick-setup commands to tell you how each bin was created.

The window called by the *Misc|High Resolution* command and the *Misc|High Bin Count* command is normally called the Performance Analyzer Setup Window. Although the Performance Analyzer Emulation Window (where the actual analysis is viewed) is called from this window (*Misc|High Resolution|Run* Pull-down and *Misc|High Bin Count|Run* Pull-down), it will be described in a section by itself, after the following description of the Performance Analyzer Setup Window.

Performance Analyzer Setup Window

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
              High Resolution Performance Analysis Setup
 Statistics    Run    Quick-setup    Misc    Edit    Add    Delete    File

Program Span: 0000-0108    Accumulate Stats: Off                  Bins:  3
Capture Span: 0000-0108
                       Setup - Capture Range Order
            Type   Bin     Range      Description

            Miss    15   0000-009F
            Mod      1   00A0-00B6   F_HLMAIN
            Mod      2   00B7-00EB   F_INNER
            Mod      3   00EC-00FC   F_WASTE
            Miss    15   00FD-FFFF

                       Ctrl-R:Resize Ctrl-V:Move
Quick Performance Analyzer setups
sLoad    SetBkpt  ViewTrc  DisAssm  MacExec  CurMod   SrcPath  RawSrc   SymGlob  SymAlph
1Help   2ResetEm 3ResetTg 4Go      5GoFrom  6GoUntil 7StepIns 8StepLin 9StepOvr 0StepTo
```

The following commands are available from the Performance Analyzer Setup Window:

Misc|High Resolution|Statistics
Misc|High Bin Count|Statistics

The *Statistics* command calls a Pull-down Menu from which you may Accumulate, Clear and View performance analyzer statistics (sample counts).

Misc|High Resolution|Statistics|Clear
Misc|High Bin Count|Statistics|Clear

The *Clear* command clears any statistics (sample counts) gathered during previous performance analyzer emulation(s).

Misc|High Resolution|Statistics|Accumulate
Misc|High Bin Count|Statistics|Accumulate

The *Accumulate* command allows you to control whether or not statistics (sample counts) from one performance analyzer emulation to the next will be accumulated. When accumulate is OFF any statistics from previous emulations are cleared before subsequent emulations. When accumulate is ON any statistics from new emulations are

added to the statistics from previous emulations. The state of the accumulate command is displayed next to the command in the Pull-down Menu and in the Performance Analyzer Setup Window just below the Menu Bar.

Misc | High Resolution | Statistics | View
Misc | High Bin Count | Statistics | View

The *View* command allows you to view the results of previous performance analyzer emulations. See the description of the Performance Analyzer Emulation Window for more information (page 7-49).

Misc | High Resolution | Run
Misc | High Bin Count | Run

The *Run* command calls a Pull-down Menu from which you may start a performance analyzer emulation. Note that a performance analyzer setup must be defined before any of these commands will function.

Misc | High Resolution | Run | Reset (emulator)
Misc | High Bin Count | Run | Reset (emulator)

The *Reset (emulator)* command starts a performance analyzer emulation from a **Reset** condition. The **Reset** signal will be supplied by the emulator itself. See the description of the Performance Analyzer Emulation Window for more information (page 7-49).

Misc | High Resolution | Run | Reset (target)
Misc | High Bin Count | Run | Reset (target)

The *Reset (target)* command starts a performance analyzer emulation from a **Reset** condition. The **Reset** signal will be supplied by the target system. Emulation will not begin until a **Reset** signal is received from the target system. See the description of the Performance Analyzer Emulation Window for more information (page 7-49).

Misc | High Resolution | Run | Go
Misc | High Bin Count | Run | Go

The *Go* command starts a performance analyzer emulation. Emulation begins at the code memory location indicated by the PC. See the description of the Performance Analyzer Emulation Window for more information (page 7-49).

Misc | High Resolution | Quick-setup
Misc | High Bin Count | Quick-setup

The *Quick-setup* command calls a Pull-down Menu from which you may to quickly define one of four types of automatic performance analyzer setups. They are N-Equal, Module, Procedure and Line Numbers. The number of Bins used for a quick-setup can also be set, otherwise the default number of bins will be used. No bins will be created with Capture Ranges out of the Capture Span used for the setup.

Misc | High Bin Count | Quick-setup | Bins

The *Bins* command allows you to specify the number of bins to use in a Quick-setup. However, if there are not enough ranges (generated by the quick setup) to fill the number of bins specified the number of bins will be be set to the number needed. For example, if you specify 12 bins and then select a Module quick setup and there are only 4 modules, the number of bins will be set to 4.

In addition, if the number of bins is not defined before a quick setup is selected, the number of bins will default to 15 for a High Resolution Performance Analyzer setup and 50 for a High Bin Count Performance Analyzer setup. Of course, this number may be automatically adjusted downward as described above. The maximum number of bins that may be specified is 15 for a High Resolution Performance Analyzer setup and 999 for a High Bin Count Performance Analyzer setup. The number of bins is displayed in the Quick-Setup Pull-down Menu and at the upper right in the Performance Analyzer Setup Window.

If the maximum number of bins is specified there is a special case where the number of bins may be decremented by one. This will happen if a Miss Bin is needed.

Misc | High Resolution | Quick-setup | N-Equal
Misc | High Bin Count | Quick-setup | N_Equal

The *N-Equal* command is used to quickly define a setup of equally sized bins. It does this by partitioning the Capture Span into N equally sized areas (where N is the number of Bins). Each created bin will be created as Type 'Neql' with the Description set with the number of bytes in the Capture Range.

Misc | High Resolution | Quick-setup | Module
Misc | High Bin Count | Quick-setup | Module

The *Module* command is used to quickly define a setup monitoring module address ranges. If possible, each bin will monitor one module; as specified by the program currently loaded into code memory. If the program contains more modules than the number of Bins, each bin may monitor several modules.

Each created bin will be created as Type 'Mod' with the Description set with the name(s) of the module(s) monitored by the bin. Each bins Capture Range will be from the starting address of the first module monitored by the bin through the ending address of the last module monitored by the bin, inclusive.

Misc | High Resolution | Quick-setup | Procedure
Misc | High Bin Count | Quick-setup | Procedure

The *Procedure* command is used to quickly define a setup to monitor the address ranges between procedures, functions and global (public) code labels. If possible, each bin will monitor the range between one procedure, function, or global code label and the next. If the program contains more procedures, functions, and global code labels than the number of Bins, each bin may monitor the address range encompassing several procedures, functions, or global code labels.

Each created bin will be created as Type 'Proc' with the Description set with the name(s) of the procedure(s), function(s), or global code label(s) monitored by the bin. Each bins Capture Range will be from the address of the first procedure, function, or global code label monitored by the bin through the address (-1) of the first procedure, function, or global code label monitored by the next bin.

Note that global code memory labels are included because not all language processors (compilers) mark procedures/functions properly in the absolute object module debug records.

Misc|High Resolution|Quick-setup|Line Numbers
Misc|High Bin Count|Quick-setup|Line Numbers

The *Line Numbers* command is used to quickly define a setup monitoring line number address ranges. If possible, each bin will monitor the range between one line number and the next line number. If the program contains more line numbers than the number of Bins, each bin may monitor the address ranges encompassing several line numbers.

Each created bin will be created as Type 'Lnum' with the Description set with the name(s) of the module(s) and line number(s) monitored by the bin. Each bins Capture Range will be from the address of the first line number monitored by the bin through the address (-1) of the first line number of the next bin.

Misc|High Resolution|Misc
Misc|High Bin Count|Misc

The *Misc* command calls a Pull-down Menu from which you may Clear a defined performance analyzer setup, change the Sort Order used to display a performance analyzer setup and to define the Capture Span the performance analyzer setup will encompass.

Misc|High Resolution|Misc|Clear
Misc|High Bin Count|Misc|Clear

The *Clear* command is used to clear all bins associated with a defined performance analyzer setup. This includes any statistics from previous performance analyzer emulations.

Misc|High Resolution|Misc|Sort Order
Misc|High Bin Count|Misc|Sort Order

The *Sort Order* command is used to change the sort order of the displayed bins in the current performance analyzer setup. The setup can be sorted by either Bin Number or Capture Range. The current sort order is displayed in the Pull-down Menu and in the title above the bins in the performance analyzer setup window.

Misc|High Resolution|Misc|Capture Span
Misc|High Bin Count|Misc|Capture Span

The *Capture Span* command allows you to specify the code memory address span over which the bin Capture Ranges may be defined. Neither Quick-setup bins or user-defined bins can have a capture range out of that span. The capture span is displayed in the upper left of the performance analyzer setup menu.

If a quick setup is selected before the capture span has been defined, the capture span will default to the address range covered by the program currently loaded in code memory. If no program is loaded the capture span will then default to all of addressable code memory.

Misc|High Resolution|Edit
Misc|High Bin Count|Edit

The *Edit* command allows you to edit the contents of the currently highlighted bin. Any bin can be edited except for the Miss Bin. All fields of a bin (Number, Capture Range and Description) can be edited except for bin Type which is changed automatically by the software depending on how the other fields are edited. In general, the bin type will be changed to 'User' if any fields of the bin are changed.

If the edited fields are valid the Miss Bins will be recalculated and the setup will be sorted and redisplayed.

Note that if the *Edit* command is selected when no bins are defined you are given an empty bin to edit, which is effectively the *Add* command (below).

Misc|High Resolution|Add
Misc|High Bin Count|Add

The *Add* command allows you to add a bin to the current performance analyzer setup. When the *Add* command is selected control is automatically passed to the Edit screen with an empty bin and edit screen rules apply. If the edited fields are valid the bin will be added to the setup, the Miss Bins will be recalculated, the number of Bins will be incremented by one and the setup will be sorted and redisplayed.

Misc|High Resolution|Delete
Misc|High Bin Count|Delete

The *Delete* command is used to delete the currently highlighted bin from the current performance analyzer setup. Any bin but the Miss Bin can be deleted. After a bin has been deleted the miss bins will be recalculated, the number of Bins will be decremented by one and the setup will be sorted and redisplayed.

Misc|High Resolution|File
Misc|High Bin Count|File

The *File* command calls a Pull-down Menu from which you may save a performance analyzer setup to a disk file and to load a previously saved performance analyzer setup from a disk file.

Misc|High Resolution|File|Save
Misc|High Bin Count|File|Save

The *Save* command allows you to save the currently defined performance analyzer setup to a disk file. You will be prompted for the name of the file. A saved setup can be restored later using the *Load* command (below).

Misc|High Resolution|File|Load
Misc|High Bin Count|File|Load

The *Load* command allows you to restore a previously stored performance analyzer setup (via the *Save* command above). You will be prompted for the name of the file.

Performance Analyzer Emulation Window

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
                   High Resolution Performance Analyzer
 Raw   Symbolic   HLL   Mixed   Counts   Expand   Misses   Write   Help

Counts:OFF  Expand:ON    Misses:ON         Resets:1       PC:00F4
Raw       Bins:16     Time:          47,648,114μs  Samples:          8,583,899
                                  RESET EMUL
 Bin Name/                                Percentages    Graphic %-in-Range   1
 Address        Bin                       This  Cumul     1 2  3 4  5 6  7 8  9 0
 Range     Bin# Type  Description         Bin   ative   04826048260482604826048260

 00A0-00B6    1 Mod   F_HLMAIN             1.5*  1.5
 00B7-00EB    2 Mod   F_INNER             36.8* 38.3
 00EC-00FC    3 Mod   F_WASTE             61.7*100.0
 .........   15 Miss  (Enabled)            0.0*100.0
 0000-009F   15
 00FD-FFFF   15

                              Esc:Break Emulation
Display Mode:  bar graph lines only (one per bin)
```

The Performance Analyzer Emulation Window is called when a performance analyzer emulation is started by the following commands:

Misc|High Resolution|Run|Reset (emulator)
Misc|High Resolution|Run|Reset (target)
Misc|High Resolution|Run|Go
Misc|High Bin Count|Run|Reset (emulator)
Misc|High Bin Count|Run|Reset (target)
Misc|High Bin Count|Run|Go

or when a post-emulation review of statistics is called by the following commands:

Misc|High Resolution|Statistics|View
Misc|High Bin Count|Statistics|View

This window shows the predefined bins along with a periodically updated display. The data (sample counts) is updated approximately once every two seconds during an emulation. From this window you can:

Display bin statistics only (*Raw*)

Display all labels and bin statistics (*Symbolic*)

Display HLL source images/lines and bin statistics (*HLL*)

Display all labels, HLL source images/lines and bin statistics (*Mixed*)

Toggle bin statistics display between bar graph lines and actual bin sample counts (*Counts*)

Toggle to/from additional information for each range in each bin (*Expand*)

Toggle to/from accumulating Miss Bin Count in total sample count and percentages (*Misses*)

Write performance analysis information to a file (*Write*)

Performance Analyzer Status Information

The top delimiter line of Performance Analyzer Emulation Window displays the type of performance analysis currently being viewed. Several status parameters are displayed in the two lines directly below the top delimiter line. They are described as follows:

Counts Indicates if the bin statistics are displayed as actual sample counts (ON) or as a percentage bar graph (OFF). See the *Counts* command (below) for more information.

Expand Displays the current Expand mode setting (ON/OFF). See the *Expand* command (below) for more information.

Misses Indicates if Misses are being added in the statistics. See the *Misses* command (below) for more information.

Resets Displays the number of resets that occurred in the current emulation cycle.

PC Displays the hexadecimal address of the next instruction which will be executed.

Brk Addr Displays the hexadecimal address at which emulation was stopped (post-emulation review only).

Bins Displays the number of bins defined in the current setup.

Time Displays the execution time of the current emulation cycle.

Samples Displays the total sample count.

Display Mode The current display mode is shown below the **Counts** status. See the *Raw*, *Symbolic*, *HLL* and *Mixed* commands (below) for more information.

Performance Analyzer Emulation Window Commands

Note that there are 8 possible paths to each of the following commands (see page 7-49), therefore, only the command will be listed and not the full command sequence.

Raw

When the *Raw* command is selected, only bin statistics are displayed (one count or bar graph line per bin).

Symbolic

When the *Symbolic* command is selected, one statistic line (count or bar graph) for each bin is displayed, followed by one line for each global (public) and/or local code memory label within each address range in that bin. Model file directive A42 (see Appendix M, Model File Configuration) controls whether only global code labels or only local code labels, or both, are displayed. In the display, global code labels are preceded by an asterisk to distinguish them from local code labels.

HLL

When the *HLL* (High Level Language) command is selected, one statistic line (count or bar graph) for each bin is displayed, followed by the HLL source line images within each address range in that bin. If a particular HLL source image is not available to the emulator only the module name and line number will be displayed. If the program loaded into code memory does not contain source line number debug

records, it is not possible for the emulator to display either HLL source line images or module names and line numbers.

Mixed

When the *Mixed* command is selected, one statistic line (count or bar graph) for each bin is displayed, followed by code labels and HLL source images. If a code memory label and an HLL source image are located at the same code memory address, the code memory label will appear in the display before the HLL source image.

Counts

When the *Counts* command is selected, it toggles the bin statistics display from showing a bar graph (percentage) line for each bin to showing the actual sample (hit) count for that bin. The Counts display command is independent of any other display control command. The sum of the sample counts for each bin usually equals the total sample count shown at the top-right of the display at **Samples**, however, the Miss Bin may or may not be included in the total **Samples** count. See the *Misses* command below for more details.

Note that an asterisk following the percentage of time spent in a bin indicates a non-zero sample count for that bin. This indication is useful, as the percentage of time spent in a particular bin may be non-zero, but less than 0.1% (in which case, the percentage of time spent in that bin is displayed as 0.0*).

Expand

When the *Expand* command is selected, it toggles the display into and out of expanded display mode. The *Expand* command is independent of any other display control command. When expanded mode is ON, following each statistic line for each bin, there will be one line showing each address range in the bin if:

1) that bin is a named bin (has a name), or

2) that bin contains more than one address range

The expanded display mode can be entered only if there is at least one bin that satisfies one of these conditions.

Misses

The *Misses* command, a toggle, controls whether or not "misses" are included in the total sample count displayed at the top-right of the screen at **Samples** and in the cumulative percentage statistic.

The performance analyzer allocates the Miss Bin to monitor program execution in code memory locations not being explicitly monitored by the user. The Miss Bin does not exist if the user is explicitly monitoring every code memory location. Even when the *Misses* Command is toggled to the OFF position (and thus, the percentage for the Miss Bin is computed as zero), the actual sample counts in the Miss Bin may be viewed by toggling **Counts** to ON, using the *Counts* command above.

Source/Symbols

When the *Source/Symbols* command is selected a Pull-down Menu of options relating to High Level Language and Symbol display and search is displayed.

```
Source/Symbols  Break/Trace  Help
┌──────────────────────────────────┐
│Module Update              AUTO ► │
│Current Module  Shift-F6        ► │
│Source Path     Shift-F7        ► │
├──────────────────────────────────┤
│Raw Source      Shift-F8        ► │
│────────────Structure─────────────│
│Modules         Ctrl-F1         ► │
│Scopes          Ctrl-F2         ► │
│Line Numbers    Ctrl-F3         ► │
│─────────────Symbols──────────────│
│Global          Shift-F9        ► │
│Local           Ctrl-F4         ► │
│Alpha           Shift-F10       ► │
│Address         Ctrl-F5         ► │
└──────────────────────────────────┘
```

Source/Symbols | Module Update

The *Module Update* command is used to select the Module Update Mode. This mode determines how the software will decide what the Current Module (see *Source/Symbols | Current Module* below) is. The mode choices are *Auto* (automatic) and *User* (user-defined).

Source/Symbols | Module Update | Auto

The *Auto* Module Update Mode means that the software will automatically determine what the Current Module is. This determination is made based on the PC address. If the PC address falls within the address range of a module, that module will be made the Current Module.

Source/Symbols | Module Update | User

The *User* Module Update Mode means that you have set (or will set) the Current Module to a specific module which will remain the Current Module until the Module Update Mode is changed to *Auto* or until you change the Current Module.

Source/Symbols | Current Module

The *Current Module* command is used to set the Current Module to a specific module. When a Current Module has been set this way the Module Update Mode is set to *User*.

The concept of the Current Module is to make Symbolic searches context sensitive. If there are two symbols with the same name (that were defined in two different modules) the Current Module will be searched first.

Source/Symbols | Source Path

The *Source Path* command is used to set the HLL (High Level Language) search path directory used to locate the source or listing files for each module in the program currently loaded into code memory. The source files (C) or listing files (PL/M) must exist in the HLL search path to enable the emulation system to locate and display HLL source images.

The HLL search path may also be set using the '-s' command line option (Appendix J).

```
Raw Source Code
Module F_HLMAIN              Language : C          File : f_hlmain.c
Line #

Line # Raw Source Code

    1  /* File:  f_hlmain.c */
    2
    3  /* C Language Demo Program 'F_DEMO.AOM' (AOM == Absolute Object Modu
    4  ** for use with the Franklin/Keil 8051 C Cross-compiler,
    5  ** Version 2.12 or later.
    6  ** Versions actually used here:        Compiler:  V3.06 (Professional E
    7  **                                    Assembler:  V4.4
        Ctrl-R:Resize Ctrl-V:Move   Tab.Shift-Tab:Select
```

The *Raw Source* command displays the raw source file for a specified module. Each source line image is preceded by its line number. The line numbers that are highlighted specially correspond to line numbers with entries in that module's line number table, meaning they may be referenced symbolically throughout the software (e.g., in setting breakpoints).

File information, such as the source language and source filename are displayed along with the name of the module. The module being displayed can be changed by typing a new module name at the Enter Module prompt.

The displayed source images can be positioned quickly to a particular line number by typing the desired line number at the Line Number prompt.

Source/Symbols | Modules

```
Modules
Enter Module

Address
  Range         Module Name          Language        Source File

0000-0000 F_VECT
00A0-00B6 F_HLMAIN                   C           f_hlmain.c
00B7-00EB F_INNER                    C           f_inner.c
00EC-00FC F_WASTE                    C           f_waste.c

             Ctrl-R:Resize Ctrl-V:Move
```

The *Modules* command displays module information for all modules. The information is displayed sorted by module address range.

The Enter Module prompt is used for a special type of symbolic search. If the entered string is found to be one of the module names the display is scrolled so that name is displayed. In addition, a full name does not have to be entered for the search to work. If only a partial name is entered the first name found that starts with the entered string will be selected.

Two screen modes are available for this command. The narrow screen contains just the module address range and name of each displayed module. The wide screen contains (in addition to the address range and name) the Source File and Language for each module. The **Ctrl-R** command is used to toggle between the screen modes.

```
                                   Scopes
 Enter Mod / Proc

 Address      Module / Procedure Name
  Range        (Number = Scope Level)          Language       Source File

 0000-0000  1 F_VECT
 00A0-00B6  1 F_HLMAIN                          C         f_hlmain.c
            2   MAIN
 00B7-00EB  1 F_INNER                           C         f_inner.c
            2   _INNERLOOP
 00EC-00FC  1 F_WASTE                           C         f_waste.c
            2   WASTETIME
                            Ctrl-R:Resize Ctrl-V:Move
```

The *Scopes* command displays scope information for all modules and procedures. The information is displayed sorted first by module address range and then alphabetically by procedure name within each module.

The Enter Module/Procedure prompt is used for a special type of symbolic search. If the entered string is found to be one of the module, or procedure names the display is scrolled so that name is displayed. In addition, a full name does not have to be entered for the search to work. If only a partial name is entered the first name found that starts with the entered string will be selected.

Two screen modes are available for this command. The narrow screen contains just the module address range and name of each displayed module. The wide screen contains (in addition to the address range and name) the Source File and Language for each module. The **Ctrl-R** command is used to toggle between the screen modes. In addition, Procedure names and their Scope Levels are listed for each module.

Source/Symbols | Line Numbers

```
                                Line Numbers
 Enter Module

 Address   Module Name
  Range              Line # Addr         HLL Source Image  /  Symbol Name

 0000-0000 F_VECT
 00A0-00B6 F_HLMAIN
                         33 00A0       state = 0;                   /* mimic ACC Po
                            00A0 MAIN
                         35 00A3       for ( counter = 0;  ;  counter++ ) {
                         36 00A5           P1_0 = 0;                /* Set P1.0 low
                         37 00A7           innerloop( 10 );         /* Generate 5 p
                           Ctrl-R:Resize Ctrl-V:Move
```

The *Line Numbers* command displays line number information for all modules. The information is displayed sorted first by module address range and then by ascending line numbers within each module.

The Enter Module prompt is used for a special type of symbolic search. If the entered string is found to be one of the module names the display is scrolled so that name is displayed. In addition, a full name does not have to be entered for the search to work. If only a partial name is entered the first name found that starts with the entered string will be selected.

Two screen modes are available for line number information for this command. The narrow screen contains the Line Number, its Address and the names of any Symbols defined at that address. The wide screen contains (in addition to the line number, address and symbol name) the Source Image for the line number. For lines describing modules, the information displayed for each module is the address range and name.

Source/Symbols | Global

```
Global Symbols - Alphabetized
 Enter Symbol

 Memory              Symbol Name            Symbol
 Space     Addr   (* Global + Predef)       Value   Data Type   Defined Within

 BIT       00D6   + AC                          0   unknown
 DIRECT    00E0   + ACC                         0   unknown
 BIT       00F8   + ALF                         0   unknown
 DIRECT    00F0   + B                           0   unknown
 BIT       00C8   + CAP2                        0   unknown
 BIT       00C9   + CNT2                        0   unknown
 BIT       00D7   + CY                          0   unknown
                  Ctrl-R:Resize Ctrl-V:Move
```

The *Global* command displays all global (public) symbols associated with the program currently loaded in code memory. Symbols are listed sorted alphabetically. See Symbol Window Information below.

Source/Symbols | Local

```
Local Symbols - By Module
 Enter Symbol

 Memory           Symbol Name          Symbol                  Defined Within
 Space     Addr   (M=Mod  P=Proc)      Value   Data Type   (Number = Scope Level)

    00A0-00B6 M F_HLMAIN
 DIRECT    0020     COUNTER                0   char        1 F_HLMAIN
 CODE      0000     _ICE_DUMMY_         ----   uint        1 F_HLMAIN
    00B7-00EB M F_INNER
 CODE      0000     _ICE_DUMMY_         ----   uint        1 F_INNER
                  P _INNERLOOP
 DIRECT    0022     I                      0   char        2 _INNERLOOP
                  Ctrl-R:Resize Ctrl-V:Move
```

The *Local* command displays all local symbols associated with the program currently loaded in code memory. Symbols are listed sorted first by module (or procedure) and then alphabetically within each module. See Symbol Window Information below.

Source/Symbols | Alpha

```
All Symbols - Alphabetized
 Enter Symbol
 Memory              Symbol Name            Symbol                    Defined Within
 Space     Addr  (* Global + Predef)        Value   Data Type   (#=Level M=Mod P=Proc)

 BIT       00D6  + AC                           0   unknown
 DIRECT    00E0  + ACC                          0   unknown
 BIT       00F8  + ALF                          0   unknown
 DIRECT    00F0  + B                            0   unknown
 BIT       00C8  + CAP2                         0   unknown
 BIT       00C9  + CNT2                         0   unknown
 DIRECT    0020    COUNTER                      0   char        1 M F_HLMAIN
                  Ctrl-R:Resize Ctrl-V:Move
```

The *Alpha* command displays all local and global (public) symbols associated with the program currently loaded in code memory. Symbols are listed sorted alphabetically. See Symbol Window Information below.

```
                              All Symbols - By Address
 Enter Symbol
 Code Xdata Direct Indirect Bit Number

 Memory           Symbol Name          Symbol                  Defined Within
 Space    Addr (* Global + Predef)     Value   Data Type   (#=Level M=Mod P=Proc)

 DIRECT   0020   COUNTER                   0  char       1 M F_HLMAIN
 DIRECT   0021 * STATE                     0  char       1 M F_HLMAIN
 DIRECT   0022   I                         0  char       2 P _INNERLOOP
 DIRECT   0023   I                         0  char       2 P WASTETIME
 DIRECT   0024   REPEAT_CNT                0  char       2 P _INNERLOOP
 DIRECT   0080 + P0                       80  unknown
                            Ctrl-R:Resize Ctrl-V:Move
```

The *Address* command displays all local and global (public) symbols associated with the program currently loaded in code memory. Symbols are listed sorted first by memory space and then by address within each memory space. The ← and → keys select the desired memory space and scroll the display to the first symbol in that memory space. If no symbols exist in a particular memory space the ← and → keys will skip that memory space. See Symbol Window Information below.

Symbol Window Information

Two screen modes are available for these commands. The narrow screen contains just the memory space, address, and name of each displayed symbol. The wide screen contains (in addition to memory space, address, and name) value, data type and scope information for each symbol. The **Ctrl-R** command is used to toggle between the screen modes.

If the symbol name is preceded by a plus sign that symbol was read from the $MODEL file during iceMASTER initialization. If the symbol name is preceded by an asterisk that symbol is defined as a global (public) symbol. Note that all symbols read from the $MODEL file are treated as global (public) symbols.

The value of a symbol can be displayed in hexadecimal, decimal, or ASCII. If the data type of the symbol is known, the display format selected is defined by Model File Directive A50, which you can modify (see Appendix M, Model File Configuration). The default display format selections are hexadecimal for char, unsigned char and float, and decimal for int, unsigned int, long and unsigned long. If the data type of a symbol is unknown the symbol value is displayed in hexadecimal. The number of bytes displayed for an unknown data type is by default 1 but can be changed using the *Configure|Options|Unknown data type size* command (page 7-22).

The scope information displayed for a symbol is dependent on the type of symbol it is. In general, the name of the module or procedure that the symbol was defined in will be displayed along with scope/nesting level information.

The *Break/Trace* command calls a Pull-down Menu which gives you direct access the the break-point and tracing features of the emulator.

```
/Symbols  Break/Trace  Help
 Set              Shift-F2  ►
 Clear
 Disable
 Enable
 ------------------------------
 Trace Trigger         End  ►
 Break-Count             0  ►
 ------------------------------
 View Trace       Shift-F3  ►
```

Break/Trace | Set

The *Set* command calls the Break Menu Window from which you may add, remove, edit or just view breaks (both Simple and Complex). If breaks are added or edited in the Break Menu, they are evaluated and set only upon exit from the Break Menu.

The Break Menu also provides access to the Opcode Class Menu Window.

The Break Menu is divided into two windows:

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
                       Break-Points & Trace-On/Off
 Add     Remove     Edit     Hex     Symbolic    Opcode_Class    Load    Save
                                   Simple
                 ADDRESS RANGE

 CBREAK          00A0 -
 TRON            0000 -



                                   Complex
                  CODE         DIRECT        BIT        OPCODE      OPCODE      IMMEDIATE
                 ADDRESS   &  ADDRESS  &  ADDRESS  &  VALUE    &  CLASS    &  OPERAND

               Ctrl-R:Resize Ctrl-V:Move  Tab,Shift-Tab:Select
Add a break element
sLoad    SetBkpt  ViewTrc  DisAssm  MacExec  CurMod  SrcPath  RawSrc  SymGlob 1SymAlph
1Help   2ResetEm 3ResetTg 4Go      5GoFrom  6GoUntil 7StepIns 8StepLin 9StepOvr 0StepTo
```

Simple Break Window

Simple Breaks are single addresses or address ranges where break-points are set or where trace is turned ON or OFF. A Code Break-Point, a Trace-On-Point (Model 400 emulators only) or a Trace-Off-Point (Model 400 emulators only) may be set for any code memory address (or range). An External Data Break-Point may be set for any External Data address (or range).

Use the **Tab** key to toggle between the Simple Break Window and the Complex Break Window.

Complex Break Window

Complex Breaks can be Code Break-Points, Trace-On-Points (Model 400 emulators only) and Trace-Off-Points (Model 400 emulators only). Each Complex Break element (one per line) is evaluated as the AND condition of a Code Address (or range), a Direct Address (or range), a Bit Address (or range), an Opcode Value (or range of values), an Immediate Operand Value (or range of values) and an Opcode Class. If any of the fields are left blank they are not ANDed during the evaluation.

Obviously, it is easy to create a Complex Break element that makes no sense (sets no breaks). For example, if an opcode value of 80h (SJMP) is selected along with any direct address this element will evaluate to NULL (no breaks). When the Complex Breaks are evaluated, NULL elements are considered errors and you will be notified of that by an error message. The Complex Break element(s) that evaluate to NULL will be marked in the Complex Break Window by replacing the break type (see CBREAK, TRON and TROFF under *Break/Trace|Set|Add* below) with the type NULL. The type will remain NULL until Complex Break elements are re-evaluated.

Use the **Tab** key to toggle between the Complex Break Window and the Simple Break Window. When the Symbolic Mode (see *Break/Trace|Set|Symbolic* on page 7-60) is active the **Ctrl-←** and **Ctrl-→** keys are used to scroll the displayed Complex Break fields (all fields cannot fit on the screen at one time).

Break/Trace|Set|Add

```
Add      Remove    Edit      Hex       Symb
CBREAK:  Code Break-Point
XBREAK:  External Data Break-Point
TRON:    Trace-On-Point
TROFF:   Trace-Off-Point
```

When the *Add* command is selected a Pull-down Menu is called from which the type of break to add may be set. Code Break-Points, Trace-On-points (Model 400 emulators only) and Trace-Off-Points (Model 400 emulators only) may be set for both Simple and Complex Break Windows. External Data Break-Points may be set in only the Simple Break Window.

Break/Trace|Set|Add|CBREAK: Code Break-Point

When the *CBREAK: Code Break-Point* command is selected the type of break element to be added is set to a Code Break-Point. Control is then immediately passed to the *Break/Trace|Set|Edit* command (below).

Break/Trace|Set|Add|XBREAK: External Data Break-Point

When the *XBREAK: External Data Break-Point* command is selected the type of break element to be added is set to an External Data Break-Point. Control is then immediately passed to the *Break/Trace|Set|Edit* command (below). This command is available only for the Simple Break window.

Break/Trace|Set|Add|TRON: Trace-On-Point

This command is available on Model 400 emulators only. When the *Trace-On-Point (TRON)* command is selected the type of break element to be added is set to a Trace-On-Point. Control is then immediately passed to the *Break/Trace|Set|Edit* command (below). Note that executing through a Trace-On-Point causes the Trace Trigger Output signal to strobe low (see Probe Clip Assembly on page 3-6).

Break/Trace|Set|Add|TROFF: Trace-Off-Point

This command is available on Model 400 emulators only. When the *TROFF: Trace-Off-Point* command is selected the type of break element to be added is set to a Trace-Off-Point. Control is then immediately passed to the *Break/Trace|Set|Edit* command (below).

Break/Trace|Set|Remove

The *Remove* command is used to remove the currently highlighted break element.

Break/Trace|Set|Edit

If the Simple Break Window is the current window when the *Edit* command is selected, a Dialog Box will prompt you for an address range for a Simple Break.

If the Complex Break Window is the current window when the *Edit* command is selected, a Pull-down Menu is called from which you can select which Complex Break field to edit a Code Address, a Direct Address, a Bit Address, an Opcode Value, an Opcode Class, or an Immediate Operand.

```
Edit    Hex    Symbolic
┌─────────────────────────┐
│Code Address Range      ►│
│Direct Address Range    ►│
│Bit Address Range       ►│
│Opcode Range            ►│
│Opcode Class            ►│
│Immediate Operand Range ►│
└─────────────────────────┘
```

Break/Trace|Set|Edit|Code Address Range

When the *Code Address Range* command is selected a Dialog Box will prompt you for a Code Address Range. Either a single address or an address range may be entered (numerically or symbolically).

Break/Trace|Set|Edit|Direct Address Range

When the *Direct Address Range* command is selected a Dialog Box will prompt you for a Direct Address Range. Either a single address or an address range may be entered (numerically or symbolically).

When Complex Breaks are evaluated, a direct address (or range) is evaluated by searching the program loaded in code memory for instructions that have direct operands and then comparing the entered address (or range) with those operands when found.

Break/Trace|Set|Edit|Bit Address Range

When the *Bit Address Range* command is selected a Dialog Box will prompt you for a Bit Address Range. Either a single address or an address range may be entered (numerically or symbolically).

When Complex Breaks are evaluated, a bit address (or range) is evaluated by searching the program loaded in code memory for instructions that have bit operands and then comparing the entered address (or range) with those operands when found.

Break/Trace|Set|Edit|Opcode Range

When the *Opcode Range* command is selected a Dialog Box will prompt you for an Opcode Range. Either a single opcode value or a range of opcode values may be entered.

opcode (or range).

Break/Trace | Set | Edit | Opcode Class

When the *Opcode Class* command is selected a Dialog Box will prompt you for an Opcode Class name (see *Break/Trace | Set | Opcode Class* below).

When Complex Breaks are evaluated and the entered opcode class is found, the program loaded in code memory is searched for opcodes that match the opcodes found in that opcode class.

Break/Trace | Set | Edit | Immediate Operand Range

When the *Immediate Operand Range* command is selected a Dialog Box will prompt you for a Immediate Operand Range. Either a single immediate operand value or a range of immediate operands values may be entered.

When Complex Breaks are evaluated, an immediate operand (or range) is evaluated by searching the program loaded in code memory for instructions that have immediate operands and then comparing the entered immediate operand (or range) with those operands when found.

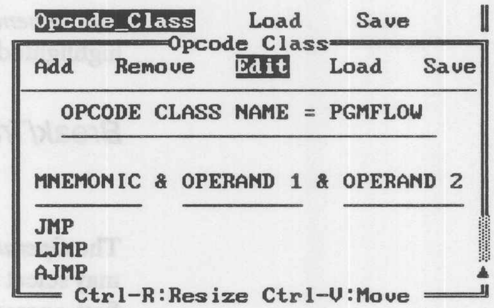
Break/Trace | Set | Hex

The *Hex* command toggles the Break Menu display mode to hex mode in which only numeric addresses and values are displayed. This mode has no effect on how addresses and values may be entered.

Break/Trace | Set | Symbolic

The *Symbolic* command toggles the Break Menu display mode to symbolic mode. In this mode addresses and values are displayed symbolically. In the Complex Break window the **Ctrl-←** and **Ctrl-→** keys are used to scroll the displayed Complex Break fields. This mode has no effect on how addresses and values may be entered.

Break/Trace|Set|Opcode Class



The *Opcode Class* command calls the Opcode Class Menu Window from which you can create or modify an opcode class. An opcode class is a user-defined grouping of instructions. Each opcode class is made of one or many opcode class elements. Each opcode class element contains three fields, the Mnemonic, Operand 1 and Operand 2. Opcode class elements can be added or removed and the fields of an opcode class element can be edited.

When an opcode class element is evaluated the three fields are effectively ANDed together. Any field left blank is not included in the AND evaluation.

Note that although the CJNE class of instructions has three operands, each CJNE instruction can be uniquely specified using the provided fields.

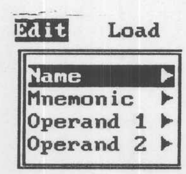
Break/Trace|Set|Opcode Class|Add

The *Add* command adds an empty Opcode Class element and automatically passes control to the *Break/Trace|Set|Opcode Class|Edit* command. The added opcode class element is then highlighted (made the current opcode class element).

Break/Trace|Set|Opcode Class|Remove

The *Remove* command removes the current (highlighted) Opcode Class element.

Break/Trace|Set|Opcode Class|Edit



When the *Edit* command is selected a Pull-down Menu is called from which you may name an Opcode Class, enter the opcode Mnemonic, and pick an operand for the Operand 1 or Operand 2 field of the currently highlighted opcode class element.

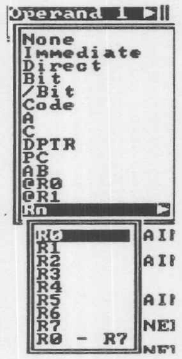
Break/Trace|Set|Opcode Class|Edit|Name

The *Name* command allows you to name an Opcode Class. The name of an opcode class is special in that when an opcode class is saved (using *Break/Trace|Set|Opcode Class|Save*), it is saved in a file name that is created automatically using the opcode class name. For example, if an opcode class is named "PGMFLOW" the file name the opcode class will be saved to is "PGMFLOW.$OC". This is true of the *Break/Trace|Set|Opcode Class|Load* command as well. When the *Load* command is selected you will be prompted for the opcode class name to load. The software will look for a file whose name is created from the opcode class name in the manner described above.

Break/Trace | Set | Opcode Class | Edit | Mnemonic

The *Mnemonic* command allows you to edit the Mnemonic field of the currently highlighted Opcode Class element. Any valid instruction mnemonic is valid.

Break/Trace | Set | Opcode Class | Edit | Operand 1



The *Operand 1* command calls a Pull-down Menu from which you may select the type of operand to use for the Operand 1 field. The first operand of the MCS-51 instruction is often called the destination operand. Note that the Operand 1 Pull-down Menu is the same as the Operand 2 Pull-down Menu.

Break/Trace | Set | Opcode Class | Edit | Operand 2

The *Operand 2* command calls a Pull-down Menu from which you may select the type of operand to use for the Operand 2 field. The second operand of the MCS-51 instruction is often called the source operand. Note that the Operand 1 Pull-down Menu is the same as the Operand 2 Pull-down Menu.

Operands

The set of choices in the Operands pull-down are:

None	no operand (used to remove an operand)
Immediate	immediate operand
Direct	direct address operand
Bit	bit address operand
/Bit	complemented contents of bit address operand
Code	code address (16-bit, 11-bit page and 8-bit relative)
A	accumulator operand (implicit)
C	carry bit operand (implicit)
DPTR	data pointer operand (implicit)
PC	PC operand (implicit)
AB	AB pair operand (implicit)
@R0	indirect register 0 operand (implicit)
@R1	indirect register 1 operand (implicit)
Rn	specific general purpose register (n = 0 through 7) (implicit)
R0 - R7	any general purpose register (implicit)

Break/Trace | Set | Opcode Class | Load

The *Load* command allows you to restore a previously saved (using *Break/Trace | Set | Opcode Class | Save*) opcode class. If an opcode class is currently loaded in memory it will be cleared before the new opcode class is Loaded. When the

Load command is selected you will be prompted for the opcode class Name to load. The software will search for the disk file for that opcode class. The file name used is created from the opcode class name. For example, if the desired opcode class is named "PGMFLOW" the file name searched for is "PGMFLOW.$OC".

Break/Trace | Set | Opcode Class | Save

The *Save* command is used to store the currently defined opcode class to a disk file that can be loaded (using *Break/Trace | Set | Opcode Class | Load*). The disk file name is created from the opcode class name. For example, if the opcode class name is "PGMFLOW" the opcode class will be saved to the file named "PGMFLOW.$OC". If the opcode class has not been named yet, you will be prompted to enter the name.

Break/Trace | Set | Load

The *Load* command allows you to restore a previously saved (using *Break/Trace | Set | Save*) break setup from a disk file. Any current break elements (Simple and Complex) are removed.

Break/Trace | Set | Save

The *Save* command allows you to store all the currently defined break elements (Simple and Complex) to a disk file. That file can then be loaded (using *Break/Trace | Set | Load*) at a later time.

Break/Trace | Clear

The *Clear* command is used to clear all breaks set through the Break Menu (*Break/Trace | Set*).

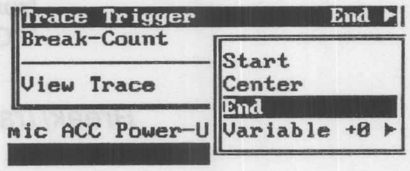
Break/Trace | Disable

The *Disable* command is used to temporarily disable all breaks set through the Break Menu (*Break/Trace | Set*). The breaks can be enabled using the *Break/Trace | Enable* command. Note that while breaks are disabled the *Break/Trace | Set*, *Break/Trace | Disable*, and *Break/Trace | Clear* commands are de-activated.

Break/Trace | Enable

The *Enable* command is used to enable breaks originally set through the Break Menu (*Break/Trace | Set*) that were temporarily disabled using the *Break/Trace | Disable* command.

Break/Trace | Trace Trigger



The *Trace Trigger* command calls a Pull-down Menu from which you may select Start, Center, End or Variable settings for the Trace Trigger. This Pull-down Menu is available on Model 400 emulators only.

The current value of the Trace Trigger is shown in the Status Window "Trig:" field and in the *Break/Trace* Pull-down Menu.

The Trace Trigger controls when emulation stops, relative to the actual break-point, in terms of the amount of information captured into the trace buffer.

Break/Trace | Trace Trigger | Start

When you select the *Start* Trace Trigger, emulation stops after approximately 4K trace frames accumulate in the trace buffer following the break-point.

Break/Trace | Trace Trigger | Center

When you select the *Center* Trace Trigger, emulation stops after approximately 2K trace frames accumulate in the trace buffer following the break-point.

Break/Trace | Trace Trigger | End

When you select the *End* Trace Trigger, emulation stops upon reaching the first break-point (i.e., zero (0) additional frames accumulate in the trace buffer following the break-point).

End is the default Trace Trigger.

Break/Trace | Trace Trigger | Variable

When you select the *Variable* Trace Trigger, you can specify exactly how many trace frames will accumulate in the trace buffer following the break-point before emulation actually stops.

Break/Trace | Break-Count

You use the *Break-Count* command to specify the number of break-points to be reached (passed through) during real-time emulation before emulation actually stops. Unlike the Repetition Counter (*Run | Repetition Count* on page 7-35), the Break-Counter is hardware controlled and operates nonintrusively during full-speed, real-time emulation.

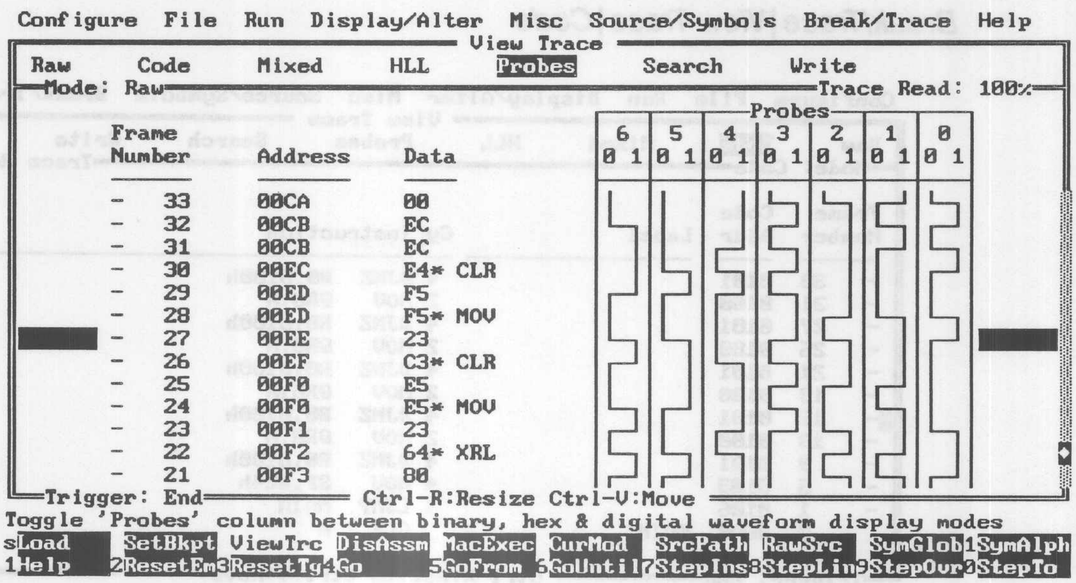
The Status Window shows the current Break-Count value in the "BrkCnt" field.

The *View Trace* command is available on Model 400 emulators only.

The *View Trace* command calls the View Trace Window from which you may examine and/or write (to a file) the contents of the trace buffer. The trace buffer can be examined/written in several different display modes: Raw, Code, HLL and Mixed.

Each frame of trace information contains the information recorded during one bus cycle. This information consists of the address (16 bits), data/opcode (8 bits), probe clip inputs (7 bits) and opcode fetch (1 bit).

Break/Trace | View Trace | Raw



7-1. (Probes - Digital Waveform, Display Mode - Raw)

The *Raw* command is used to set the Raw display mode. In the Raw display mode, you see the binary content of the address/data bus for each bus cycle (trace frame) in a raw hex format.

The fields in the Raw mode display are:

Frame Number	the relative trace buffer frame number where the frame at the trigger point is numbered zero (0)
Address	the captured address value
Data	the captured data or opcode value
Probes	the captured Trace Input signal values (see Probe Clip Assembly) where a value of zero (0) reflects a low signal and a value of one (1) reflects a high signal

The notations used in the Raw mode display are:

*	opcode-fetch cycle (beginning of instruction's execution) where the opcode mnemonic is displayed to the right of the asterisk
**(LCALL)	interrupt (1st ALE cycle)
**	interrupt (2nd
R	Reset
DMA	DMA activity
?	same as "(uncertain)" in the Code display mode
< <	(Probe column) marks next instruction to be executed (not executed yet)

Break/Trace | View Trace | Code

```
 Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
 ==================================== View Trace ==============================
  Raw      Code      Mixed      HLL      Probes      Search      Write
 =Mode: Code=================================================Trace Read: 100%=

  Frame    Code
  Number   Addr   Label                Cy Instruction                  Probes
  -------------------------------------------------------------------------
  -     33  0101                        4 DJNZ    R0,0100h              1111111
  -     31  0100                        2 MOV     @R0,A                 1111111
  -     27  0101                        4 DJNZ    R0,0100h              1111111
  -     25  0100                        2 MOV     @R0,A                 1111111
  -     21  0101                        4 DJNZ    R0,0100h              1111111
  -     19  0100                        2 MOV     @R0,A                 1111111
  -     15  0101                        4 DJNZ    R0,0100h              1111111
  -     13  0100                        2 MOV     @R0,A                 1111111
  -      9  0101                        4 DJNZ    R0,0100h              1111111
  -      5  0103                        4 MOV     SP,#24h               1111111
  -      1  0106                        4 LJMP    MAIN                  1111111
         0  00A0  MAIN:                   CLR     A                     <<<<<<<
 =Trigger: End=========== Ctrl-R:Resize Ctrl-V:Move ===========================
 Code display mode:  show trace frame content as disassembled instructions
 sLoad  SetBkpt ViewTrc DisAssm MacExec CurMod  SrcPath RawSrc  SymGlob1SymAlph
 1Help  2ResetEm3ResetTu4Go     5GoFrom 6GoUntil7StepIns8StepLin9StepOvr0StepTo
```

7-2. (Probes - Binary, Display Mode - Code)

The *Code* command is used to set the Code display mode. In the Code display mode, you see the content of the trace buffer interpreted as fully disassembled instructions, including all available symbolic information (except HLL source images).

The fields in the Code mode display are:

Frame Number	the relative trace buffer frame number where the frame at the trigger point is numbered zero (0) and the frame number displayed here is that of the last frame in the trace of the instruction's execution
Code Addr	the captured address value in the first frame (first cycle) of the trace of the instruction's execution
Label	the label(s) for the instruction at address Code Addr (if there is more than one label at this address and they won't all fit in the Label field, iceMASTER displays the first label at this address with preference given to global (public) labels) preceded by a plus sign to indicate that there is more than one label at this address

Cy	the number of cycles in the instruction's execution where this is the same as the number of frames laid down in the trace buffer for the instruction
Instruction	the disassembled instruction at address Code Addr
Probes	the captured Trace Input signal values (see Probe Clip Assembly on page 3-6) where a value of zero (0) reflects a low signal and a value of one (1) reflects a high signal

The notations used in the Code display mode are:

*LCALL	interrupt
R	Reset (only when in middle of an instruction)
DMA Cycle	DMA activity (not part of instruction's execution)
(uncertain)	(unable to decode with absolute certainty because only part of an instruction's execution trace was captured in the buffer due to automatic wraparound in the circular trace memory in the emulator)
<<<<<<	(Probe column) marks next instruction to be executed (not executed yet)

Break/Trace|View Trace|Mixed

The *Mixed* command is used to set the Mixed display mode. The Mixed display mode is a combination of the Code and HLL display modes. The HLL source images (HLL mode display), if available, appear interspersed appropriately with the generated machine code as in the Code mode display. An example of the Mixed display mode is shown under the description of the *Search* command (Figure 3).

Break/Trace|View Trace|HLL

The *HLL* command is used to set the HLL display mode. In the HLL display mode, you see the decoded content of the the trace buffer interpreted as HLL source language images. This can be done only if the iceMASTER system has access to the source/listing files corresponding to the modules in your program. Source languages supported include C and PL/M-51.

Break/Trace|View Trace|Probes

The *Probes* command cyclically toggles the display of the captured probe clip (see Probe Clip Assembly) bit values through three display modes:

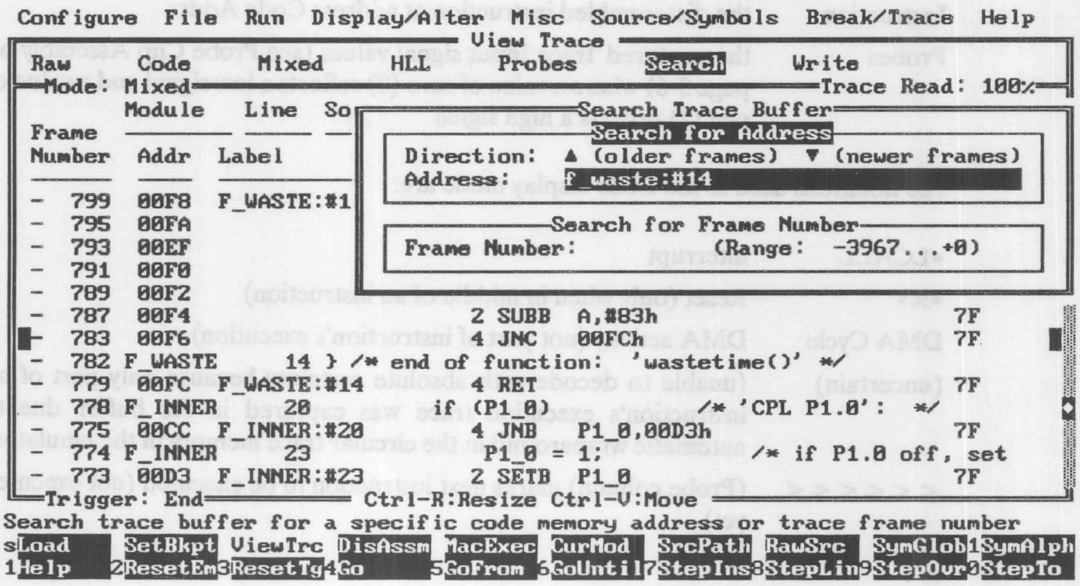
Binary (see Figure 7-2)

Hexadecimal (see Figure 7-3)

Digital waveform (see Figure 7-1)

The *Probes* command and its associated display are available in the Raw, Code and Mixed primary display modes.

The digital waveform display in the Raw display mode reflects every frame in the trace buffer. However, the waveform display in Code and Mixed modes reflects only the probe clip values in the first CPU cycle of each instruction's execution. Therefore, if a probe signal transitions during the CPU cycles of an instruction's execution, the displayed waveform may appear "broken". When you see this, you can switch into Raw display mode to get the complete picture.



7-3. *(Probes - Hexadecimal, Display Mode - HLL)*

The *Search* command allows you to search the trace buffer for a particular address, label, source line number or trace frame.

In searching for an address, label or source line number, the search commences from the current position in the trace buffer (the highlighted line in the middle of the window). You can search backwards in the trace buffer, looking for older trace frames, or forwards, looking for newer (more recent) trace frames. The ←, →, ↑ and ↓ keys are used to select which type of search is performed, and in which direction.

Break/Trace | View Trace | Write

You can write the entire trace buffer to a disk file by selecting the *Write* command. The format of the information written to that file will be just as you see it on the screen (i.e., in human-readable form, in the current display mode).

Chapter 8: Run-Time Considerations

The iceMASTER emulator is designed to be as close as is possible to an actual device. In most cases you will not be aware of any difference since the devices used to emulate are the actual microcontrollers themselves. Since we do need to know what is going on during emulation there are a few constraints placed upon us, and a few precautions you can take to prevent problems.

Static

Perhaps the most difficult problem anyone who uses MOS devices will face is static. You may go for years with no fault traceable to static, or you may blow every part you touch. The iceMASTER emulator can be as sensitive to static as any other circuit. The microcontroller devices in the probe cards are especially vulnerable since adding extra protection would change response characteristics. This would be a step away from the real world. The built-in protections internal to the devices are operative.

We do still recommend, however, that you take every precaution regarding static. The use of grounding straps, static free workstations, and a little extra care in handling the emulator (and any MOS part) can prevent troubles later.

Power

When starting an emulation session, always turn the emulator power on first, then apply power to your target system. This will ensure that any interface devices are initialized properly. If you need to test a power-up initialization routine in your target, power up the emulator first, load your program if necessary, begin execution of your program using the *Run|Reset|Target* command (page 7-33) which will cause the emulator to wait for a reset to arrive from the target, then power up your target and be sure that the power-up sequence includes a valid reset after the target is initialized (which is a good design practice anyway). If you are working in a target application which uses less than +5V, this is acceptable if:

1) The microcontroller on the probe card can be powered from the target system, and

2) The voltage levels on the signals which we monitor meet minimum TTL levels. Note that the emulator base monitors port 0, port 2, XTAL 1, #RD, #WR, ALE, and #PSEN.

Please remember that we provide power for the microcontroller on the probe card (unless a jumper allows the user to connect the VCC line to the target power pin(s)). Therefore, you should be sure the two VCC levels are close enough that the guaranteed logic high and low values are met. It is also necessary that you do not exceed the input pin voltage ratings of the devices on the probe card which allow communication with the base. We recommend that power be maintained at +5V +/- 5%. If you have any questions about a specific application, please contact us (page 1-1).

/RD And /WR Control Signals

In the 8051 family, the port register bits P3.6 and P3.7 (which correspond with the /RD and /WR control signals) must be set high in order to function normally. This is a device characteristic. The iceMASTER emulator monitors these lines to obtain data about MOVX instruction executions. If you are using either pin as a port pin, or if you allow other devices to pull either pin low independent of the microcontroller, be sure this does not happen concurrent with a MOVX instruction. If both /RD and /WR are active low during a MOVX, the emulator base may exhibit abnormal behavior.

Some probe cards provide jumpers to hold the /RD and or /WR monitor lines to the base high in these cases. If yours does not, simply be sure that the MOVX can not occur at a time that will allow both of the /RD and /WR lines to be low at the same time.

Clock Drivers

Many of the external clock drivers commonly used provide TTL level outputs. This can be a problem for not only the emulator but your target design as well. Most CMOS processors require a CMOS oscillator or clock levels to function reliably.

Do not use NMOS or HMOS parts to replace the microcontrollers on probe cards not designed for them. They are not guaranteed to work and doing so will void your warranty.

In the 8051 family, some NMOS processors had clock circuits that required XTAL 2 to be driven and XTAL 1 to be grounded. Most CMOS processors require XTAL 1 to be driven and XTAL 2 to be left floating. This presents a problem when using a CMOS part in an application designed for an NMOS part. Please refer to Chapter 4, the Probe Card Reference for settings (if available) to compensate for this. This is only a problem when using an external clock driver, as crystals are usually symmetrical enough that it doesn't matter.

Timer Values

The iceMASTER emulator automatically turns off all timer/counters when emulation stops. If a timer/counter is running when a breakpoint is encountered, it is stopped several machine cycles after the breakpoint. This characteristic can cause edge-sensitive routines to "malfunction" if breakpoints are set such that emulation stops before the edge is active. This applies to timer functions and edge-triggered interrupts. The work around: don't single-step or set breakpoints through instructions that look for or generate the edges.

If a timer/counter is turned on before beginning an emulation for the first time (by setting the appropriate bits in the TCON register using the *Display/Alter|Var/Reg* command page 7-40), the timer/counter will actually be turned on several machine cycles before the first instruction is executed.

If a timer/counter is turned on for an emulation which must be restarted (i.e., execution has already stopped at a breakpoint), the timer/counter will be turned on several machine cycles before full speed emulation begins again.

Port Registers

The values for the ports which are displayed via the SFR displays represent the actual values at the port pins and not the value in the port registers. If the value of one of the output pins is to be changed, care must be taken to ensure that a 1 is written to any input pins in the same port (otherwise they will become output pins).

Idle and Power Down Modes

During emulation, the status message

Probe Card Processor Inactive

may be displayed. This status message indicates that there is **NO ALE** signal reaching the emulator base from the probe card. Normally this will be due to the entry of **Idle** mode or **Power Down** mode on CMOS devices. If this condition occurs for any other reason, see Chapter 9, Troubleshooting.

If a device is in Idle or Power Down mode, and you execute a Host-break (i.e., manually breaking (stopping) emulation by pressing **Esc**), a Confirmation Box will prompt you with the following:

Processor not running, do you want to RESET to break emulation?

A Y response will reset the system and emulation will break at address 0. A **N** or **ESC** response will leave the device in its current state. Breaking emulation by pressing the break button on the iceMASTER emulator base has the same effect.

After successfully entering the Idle or Power Down state, no further instructions are executed so no preset breakpoints will be encountered. ALE is high during Idle and low during Power Down. The iceMASTER emulator can be put into the CMOS reduced-power modes either under program control or (in most microcontrollers) by modifying the appropriate register (using the interactive capability of the register window or through the *Display/Alter | Var/Reg* command on page 7-40). In order to invoke these modes of operation, the probe card must support the CMOS versions of its respective controllers.

Any interrupt request or a hardware reset will bring the emulator out of Idle mode. If the Idle mode is terminated with a hardware reset, the contents of the SFR's and GPR's will be lost. The communication link between the emulator and the Host Computer will also have to be re-established.

When in Power Down mode, the emulator must be reset in order to regain the communication link. For instance, if Power Down is invoked and then the Host-break function is executed (**Esc** key is pressed), you will be asked whether or not a hardware reset is to be performed. It is not mandatory to reply **Y**. It is, however, the quickest way to re-establish the communications link and resume emulation.

Microcontroller Serial Port

After breaking emulation (by breakpoint or Host-break), all interrupts are disabled. A serial port transmit or receive interrupt which occurs after the breakpoint has been encountered will be ignored.

Whether the Host Software writes to SBUF via the *Display/Alter | Var/Regs* command (page 7-40) or the program code writes to SBUF (e.g, MOV SBUF,A) the register window will not display the value written to SBUF. This is because a write operation to SBUF writes the value to the serial port transmit buffer and the read SBUF (to display the value) operation reads the serial port receive buffer (they are separate).

Watchdog Timer

For all devices with a Watchdog Timer, in order for the Reset Count in the Main Status Window to accurately reflect the number of watchdog resets, the oscillator clock frequency must be 16MHz or less. At greater than 16MHz, the emulator base will begin to miss watchdog resets and the Reset Count will be wrong, although the device will actually perform the reset and the trace buffer will reflect the event. If a watchdog reset should happen just after a break condition, before the emulator can refresh the WDT registers, there is a small possibility that the emulator base could get out of synchronization with the device. The values in the SFR's may be displayed incorrectly, and the trace buffer data may be corrupted. To re-synchronize the emulator base and the device the user must start the next emulation with a reset.

If you have any further questions about emulators we will be happy to discuss your application.

Chapter 9: Troubleshooting

Before starting any fault investigation, remove the emulator from the target system and configure the probe card for stand-alone operation. Make sure that all connectors are in good condition and fully seated. Take note of any physical damage to the unit.

Several common problems are covered in this chapter. If this isn't enough to get the emulator back into operation, contact MetaLink (see page 1-1).

With your knowledge of the application and MetaLink's knowledge of the emulator, many problems can be diagnosed within a few minutes.

Before Calling

Have the system near the phone so that problems may be "walked through". If the unit needs to be sent in for repair you will also need the following information:

1) the emulator's serial number (on the bottom of the emulator base unit)
2) the software revision level (see *Configure|Identification* on page 7-20)
3) the address to which MetaLink will ship the repaired unit

Power Indicator LED Is Not Lit

If the power indicator LED is not lit, check the usual background details, such as that the power switch is ON, the power is connected to the unit and the AC circuit breaker is ON. If these items are in order, make sure that +5V is delivered to the emulator's power connector by checking the output of the +5V power supply. Examine the emulator power connector drawing in the Hardware Installation chapter to determine the pin-outs for the male DIN connector.

Using a voltmeter, check the terminals of the power supply. If the output is correct, check the +5V at the probe card (with the emulator power switch ON). The easiest place to read this is across the large power filter capacitor on the probe card. If this reading is not +5V (+/- 5%), the problem may be in either your emulator or in your power supply. If you have another power supply capable of providing +5V at 1.5 Amps (+/- 5%), try it. If this does not solve the problem, the emulator needs to be repaired. Contact MetaLink (see page 1-1).

Active Indicator LED Is Not Lit

If the Active LED is not lit, make sure that the probe card's crystal jumpers are set correctly for the application. Refer to the Probe Card Reference for your probe card. The probe card may be configured to support clock drivers for either CMOS or NMOS controllers, regardless of the technology of the microcontroller on the probe card. Make sure that the jumper blocks have not been damaged and are making good contact at their respective posts.

Once the crystal jumper block settings have been verified as correct, make sure that the Power Down mode or Idle mode of operation (CMOS controllers only) was not inadvertently invoked through software. A Reset is the only way to terminate the Power Down mode. If the Active LED stays unlighted, contact MetaLink (see page 1-1).

Communications Failure

Make sure that the oscillator signal is present. If a crystal is used, the crystal itself may be damaged. Replace the crystal with an equivalent crystal if the no-signal condition remains. If the oscillator is present, check the probe card cable at its mating connectors.

Also check the cable for damage which may have introduced a fault in the signal activity. Replace the cable with an equivalent if flaws are seen. Refer to Chapter 3, Hardware Installation for details on the cable construction.

Next, compare the RS-232 cable with the illustration in the Hardware Installation chapter. Replace the cable if needed. A "break-out box" will facilitate a check on the presence and level of the RS-232 signals on the cable. Active signals will be at a nominal 10V and the polarity may be plus or minus depending on the state the of the hardware. If no activity is seen on Pin 2 (Transmit) the Host Computer has a fault in its interface card. If Pin 2 is active and Pin 3 is inactive (not toggling) the emulator has a fault. A persistent fault may indicate that the probe card's microcontroller has been damaged. Replace the controller with an equivalent. If the fault remains, contact MetaLink (see page 1-1).

Emulation Problems

In stand-alone mode, load and run the program DEMO.DBG that is supplied on the distribution diskettes. If the program runs correctly, the problem may not be with the emulator but with the emulator target interface. Keep an open mind. Even in known good target systems, failures occur. It may even be possible for a "real" device to work in your target where the emulator has trouble. This is usually a problem with tolerances, not differences. If you encounter this, please call us so we can determine quickly where the problem lies.

Carefully consider the application:

1) verify that the emulator is configured properly (see Chapter 4, the Probe Card Reference)
2) verify that mapping is set properly
3) look for unexpected resets (e.g., watchdog timers)
4) check interrupt routines for proper returns to normal code execution

If these procedures restore operation in the stand-alone mode, or if the unit worked in the stand-alone mode without correction, it is necessary to determine if the target is causing the fault or if the emulator has a fault that doesn't manifest in stand-alone mode.

Troubles to watch for include:

1) Watchdog Timer resets during BREAK condition
2) bus contention
3) excessive loading
4) failed components in the target system (especially failures that are likely to damage the probe card or emulator base)

If you have any questions contact MetaLink for technical assistance (see page 1-1).

Appendix A: Tutorial

This tutorial will help you become familiar with the operation of MetaLink emulators. This is an introduction and is by no means an in-depth explanation of how the iceMASTER emulator functions, nor does it fully explain its capabilities. Once you have completed the tutorial you will be ready to begin experimenting on your own and with the help of this manual to be able to understand and master the emulator.

Before You Begin

Make sure the software and hardware is installed (see Chapter 3, Hardware Installation and Chapter 5, Software Installation) and then read the Software Guide (Chapter 6) to familiarize yourself with the layout of the screen and the terminology used to describe the software system.

This tutorial assumes the emulator is functioning and is at hand ready-to-use. Throughout this tutorial we will refer to the iceMASTER emulator simply as the emulator. The iceMASTER Host Software is a windowed interface which allows access to most data directly from the Main Menu. Most windows can be resized, repositioned, activated or inactivated at your discretion. The Software Guide (Chapter 6) and Command Reference (Chapter 7) contain details on how to make these changes. For now, let's get right to some useful activities.

Begin

Begin by changing to the drive and directory in which you have installed the Host Software (we will assume the installation procedure default drive and directory, C:\IM51), as follows:

C:
CD \IM51

You are now ready to invoke the Host Software, as follows:

ICE

The initialization screen will appear and list the activities being performed as initialization proceeds. When this process is complete, the Main Menu Windows will be displayed, but since communication between the emulator and the Host Software has not been established no data will be displayed in the windows. In addition, since communication has not been established, several levels of Pull-down menus will be invoked so that the currently highlighted command is *Configure|Emulator|Execute* (see page 7-2), which is the command used to establish communication between the emulator and the Host Software.

The files DEMO.DBG and F_DEMO.AOM will be used in this tutorial session (DEMO_751.DBG for the 83C751 and 83C752). DEMO.DBG (or DEMO_751.DBG) contains full symbolic information as generated by MetaLink's 'ASM51' absolute macro cross-assembler with the $DEBUG directive set.

F_DEMO.AOM contains full, extended symbolic and source-level information as generated by Franklin's 'C-51' compiler. Rather than describing the emulator, let's run it. Learning by doing is often the shortest path to learning a new skill (or a new applications program).

Establish Host-to-Emulator Interactive Communication

Before establishing communication you need to verify the communication port selected and the chip mode of operation. The communication port listed in the Pull-down Menu next to the *Configure|Emulator|Comm port* command (page 7-3) should be the communication port installed in the Host Computer. The chip mode of operation listed in the Pull-down Menu next to the *Configure|Emulator|Mode* command (page 7-2) should be the default for your microcontroller, as specified in the Probe Card Reference (Chapter 4).

To establish communication, select the *Configure|Emulator|Execute* command. As communication is established various Status Boxes will be displayed to inform you of the operations that are taking place. When the Pull-down Menus are popped (removed from the screen) and the Main Screen Windows are repainted (with all data) we are ready to go.

Load A File Into The Emulator

We are now ready to load a file into the emulator. The files we will use for this demonstration are:

DEMO.DBG DEMO.DBG is the load file from an assembly language program which contains full symbolic information as generated by MetaLink's ASM51 absolute macro cross-assembler (using the $DEBUG directive). Note that the Tutorial description will specify DEMO.DBG in places, but if you are using an 83C751 or 83C752 you should substitute DEMO_751.DBG.

F_DEMO.AOM F_DEMO.AOM is the load file from a C program which contains full, extended symbolic and source-level information as generated by Franklin's C-51 compiler.

To load a file, select the *File* Pull-down Menu and then select the *Load* command (page 7-27). Note that we will sometimes refer to a sequence of commands by separating each command by a vertical bar, in this case *File|Load*.

A Filename Dialog Box will then prompt you for the name of the file to be loaded. Note that Filename Dialog Box includes a directory listing capability. Let's use it to find the name of the file you will be loading. At the file name prompt press ?. A Dialog Box will then prompt you for the drive and path to list a directory for. To select the current directory simply press **Enter**. A directory listing will be displayed (the same information produced by the DOS 'DIR /W/P' command). The file DEMO.DBG should be one of the files listed.

Note that pressing ! (rather than ?) will result in the directory listing complete with date/time stamp and file size information (the same information produced by the DOS 'DIR /P' command).

The Host Software assumes that the file is in the current directory unless a path is specified. Path names and filenames follow DOS conventions. After the directory is listed, the message

Press any key to continue

is displayed. Press a key and then enter the file name at the prompt. A Confirmation Box will then ask

Merge into current application environment?

Respond Y to retain the current application environment, including breakpoint settings, trace ON/OFF points and all symbolic information. Respond N to clear the current application environment before loading a new file.

For our situation, respond N. Various Status Boxes will be displayed to inform you of the progress of the *Load* while the file is loading.

The Quick Help line at the bottom of the screen shows the following status:

Read Address: xxxx -- offset into file of current code image record being read from file.

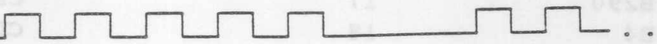
Load Address: xxxx -- address of first byte in block being written to emulator base unit.

Demo Program Overview

Before going further, let's take a look at the programs we are going to be working with. The purpose of the programs is to output a pulse train on Port 1, Pin 0 (P1.0). The pulse train is a continual repetition of:

1) a series of 5 positive going pulses of equal duration
2) a skipped pulse

The pulse train appears as follows:



There are three control loops in these programs. They are:

WASTETIME WASTETIME executes ninety (90) bus (ALE) cycles before exiting.

INNERLOOP INNERLOOP is responsible for generating the 5 pulses. It accomplishes this by calling the WASTETIME routine to generate the pulse width and spacing. It then changes the polarity of the output pin. The accumulator in DEMO.DBG and the variable 'STATE' in F_DEMO.AOM are used to reflect the state of the output pin.

OUTERLOOP OUTERLOOP is a simple, infinite loop. It uses INNERLOOP to generate the 5 pulses and then calls WASTETIME twice to generate the blank pulse between sets of 5 pulses. Notice that the Data Pointer (DPTR) in DEMO.DBG or the variable 'COUNTER' in F_DEMO.AOM is used to count the number of transmitted sets of 5 pulses.

```
                     $debug
                     $nopaging
  0090               outbit          BIT     90h
0064                                 DSEG AT 64h
0064                 tempcount:      DS      1
----                                 CSEG
0000 020030                          LJMP    start
0030                                 CSEG AT 30h
0030 900000          start:          MOV     DPTR,#0
0033 C290            outerloop:      CLR     outbit
0035 75640A                          MOV     tempcount,#10
0038 120050          innerloop:      CALL    wastetime
003B B290                            CPL     outbit
003D E4                              CLR     A
003E 309001                          JNB     outbit,skipover
0041 F4                              CPL     A
0042 D564F3          skipover:       DJNZ    tempcount,innerloop
0045 A3                              INC     DPTR
0046 120050                          CALL    wastetime
0049 120050                          CALL    wastetime
004C 80E5                            JMP     outerloop
004E 80E0            endofprogram:   JMP     start
                     ;
                     ;
0050 78FF            wastetime:      MOV     R0,#0FFh
0052 D8FE                            DJNZ    R0,$
0054 22                              RET
                                     END
```

Figure A-1. DEMO.LST

Now that a file is loaded in the emulator's code memory, we are ready to begin an emulation session. This is monitored and controlled from the Main Menu.

Let's start by running the program by selecting the *Run | Reset | Emulator* command (page 7-33). When this command is selected, the emulator base unit will reset the device in the probe card (and execution of the DEMO.DBG program will begin). The message RESET is displayed in the Status Window, to show how emulation was started. This message will be displayed until emulation is halted, at which time the reason for the break will be displayed. Also, notice that the PC Address and Execution Time are updated approximately every 1.5 seconds to indicate emulation activity.

Since there are no breakpoints currently set, press the **Esc** key to stop emulation. Note the changes in the Main Screen windows.

The SFR's (Special Function Registers) and GPR's (General Purpose Registers) are updated to reflect the values present in these registers at the instant emulation was stopped.

The PC (Program Counter), Break Address and DPTR (Data Pointer) located in the Status Window are updated to reflect current values. In this case, the Break Address will be the same as the PC.

The next instruction to be executed is highlighted in the Source Window, along with the value of any indirect operands for that instruction.

Pressing **Esc** during emulation forces a break on the emulator. This is called a Host-break. This is just one way that the user can stop emulation.

Setting Breakpoints

An iceMASTER emulator has four ways to break emulation:

1) set a breakpoint using the Break Menu (*Break/Trace|Set*, page 7-57)
2) pressing the **Esc** key (Host-break during emulation)
3) pressing the Break push-button on the emulator base
4) using the Break Input probe clip on the probe card

Let's set a simple breakpoint first. Select the *Break/Trace* Pull-down Menu and then select the *Set* command. The Breakpoint Menu will be displayed. The Breakpoint Menu is divided into two parts, simple breaks and complex breaks. The heading **Simple** should be highlighted. To move between Simple and Complex Windows use the **Tab** or **Shift-Tab** keys. From the Simple Break Window you may:

1) set a code breakpoint (or range of breakpoints)
2) set an external data breakpoint (or range of breakpoints)
3) set a trace ON or OFF point (or range of points) (Model 400 emulators only)

Select *Add*. A Pull-down Menu will show four choices (page 7-58):

1) CBREAK: Code Break-Point
2) XBREAK: External Data Break-Point
3) TRON: Trace-On-Point
4) TROFF: Trace-Off-Point

Note that the *TRON: Trace-On-Point* and *TROFF: Trace-Off-Point* commands will be de-emphasized and unavailable if the emulator is not a Model 400. The *XBREAK: External Data Break-Point* command will be de-emphasized and unavailable if the device being used does not have access external data memory (e.g., the 83C751 and 83C752).

Select the *CBREAK: Code Break-Point* command. A Dialog Box will prompt you for the address range. Enter **0** at the start address prompt. The cursor then automatically advances to the end address prompt. An entry at this prompt will specify a range of breakpoints, start address through end address, inclusive. Press the **Enter** key so that the break will be set at just address 0.

Repeat the *Add|CBREAK: Code Break-Point* command sequence and enter the next code breakpoint symbolically. Enter **WASTETIME** at the start address prompt and press **Enter** at the end address prompt. Note that you can toggle the display mode of the Break Menu to symbolic mode (break addresses displayed symbolically) by selecting the *Symbolic* command (page 7-60) and to hex mode (break addresses displayed in hexadecimal) by selecting the *Hex* command (page 7-60).

Now that we have set some breakpoints press **Esc** to return to the Main Menu and select the *Run|Reset|Emulator* command sequence (page 7-33) to begin emulation. The breakpoint at location 0 will be encountered immediately. The Status Window should show that the PC and the Break Address are both 0. Now select the *Run|Go* command (page 7-33) to resume execution from the current PC. In a few

moments, the breakpoint at WASTETIME is reached and emulation will stop. Now select the *Run | Go* command sequence once more. Now let's have a look at the Trace Buffer (Model 400 emulators only).

Viewing The Trace Buffer (Model 400 Emulators only)

Trace is always enabled at the start of emulation and trace information will always be captured unless you turn trace OFF using the *Break/Trace | Set | Add | TROFF: Trace-Off-Point* command (page 7-58). Therefore, it is not necessary to turn trace ON at every point where you may want to begin emulation.

You may view the data captured in the trace buffer by selecting the *Break/Trace | View Trace* command (page 7-65). Frame number 0 should show the address at location **WASTETIME** (50h, or 4Dh if using DEMO_751.DBG) in the trace buffer is the breakpoint. The default trigger point (END) is at the end of the stored data. Therefore, all trace data is behind (earlier than) the trigger.

Let's select the *Search* command (page 7-68) to search for a specific frame in the trace buffer. In the Search Dialog Box, use the ↓ key to highlight the frame number field and then enter **-3967**. This will cause a search for the trace frame 3967 frames prior to the trigger point at frame number 0. The Trace Buffer repaints the display to show the specified frame (on opcode fetch boundaries), or the next best thing. In this case,the next best thing is the top of the Trace Buffer at about frame number -1043.

Let's select the *Search* command again and search for a specific address in the trace buffer. In the Search Dialog Box highlight the address field and then enter **INNERLOOP**. Note that the ← and → keys allow you to toggle between the forward and backward search directions. There will be a slight delay as the software searches ahead in the Trace for the first occurrence of the symbol **INNERLOOP**. In this case, the label is found at frame number -0001.

To move around in the trace buffer the **Home** key takes you to the top (start) of the data, the **End** key takes you to the bottom (end) of the data, and the **PgUp** and **PgDn** keys move up or down (respectively) one page of data. Try it.

Select the *Raw* command (page 7-66) to change the display mode of the trace buffer from Code Mode (disassembled instructions) to Raw Mode. Trace data is now displayed in raw hexadecimal form, with instruction fetches marked by an asterisk and the mnemonic opcode. There is one trace frame for each ALE cycle, and each of these frames is displayed in the Raw Display Mode. Now select the *Code* command (page 7-65) to return to the Code Mode. Press the **Esc** key to return to the Main Menu.

```
/* File:  f_hlmain.c */

/* C Language Demo Program 'F_DEMO.AOM' (AOM == Absolute Object Mod-
ule),
** for use with the Franklin/Keil 8051 C Cross-compiler,
** Version 2.12 or later.
** Versions actually used here:     Compiler:  V3.06 (Professional Edi-
tion)
**                                 Assembler:  V4.4
**                                    Linker:  V2.7
**
** This program replicates, as closely as possible (except for actual
timing),
** the functionality of the ASM51 demo program 'demo.asm'.
**
** Modules (Files) in F_DEMO.AOM:
**   f_vect.a51 -- Assembly Language:  reserves space for interrupt
vectors
**   f_hlmain.c -- C, main program (function 'main()')
**   f_inner.c  -- C, function 'innerloop( repeat_cnt )'
**   f_waste.c  -- C, function 'wastetime()'
*/

#include <stdio.h>

sbit P1_0 = 0x90;       /* P1.0 */

/* external function prototypes (interface specifications) */
extern void innerloop(char);
extern void wastetime(void);

char   state /*= 0*/;   /* 'state' reflects value in ACC (in original
demo.asm) */
static char counter;    /* dummy counter (rolls over at 256) */

void main()
{
    state = 0;  /* mimic ACC Power-Up Reset condition (value) */

    for ( counter = 0;  ;  counter++ ) {
        P1_0 = 0;           /* Set P1.0 low at start */
        innerloop( 10 );    /* Generate 5 pulses */
        wastetime();        /* Call 'wastetime' twice to generate   */
        wastetime();        /* blank pulse between sets of 5 pulses */
    } /* end of:  for 'counter' */

} /* end of function:  'main()' */
```

Figure A-2. F_HLMAIN.LST

```
/* File:  f_inner.c */

#include <stdio.h>

/* external function prototypes (interface specifications) */
extern void innerloop(char);
extern void wastetime(void);

/* external variables */
extern char state; /* 'state' reflects value in ACC (in original
demo.asm) */

sbit P1_0 = 0x90; /* P1.0 */

void innerloop( char repeat_cnt )
{
    static char i;

    for ( i = 0;  i  repeat_cnt;  i++ ) {
        wastetime();
        if (P1_0)       /* 'CPL P1.0':  */
            P1_0 = 0; /* if P1.0 on,  clear P1.0 */
        else
            P1_0 = 1; /* if P1.0 off, set   P1.0 */
        state = 0;
        if ( P1_0 )
            state = (state) ? 0 : 1;     /* 'CPL A' */
    } /* end of:  for 'i' */

} /* end of function:  'innerloop( repeat_cnt )' */
```

Figure A-3. F_INNER.LST

```
/* File:  f_waste.c */

/* external function prototypes (interface specifications) */
extern void wastetime(void);

void wastetime( void )
{
    static char i;

    for ( i = 0;  i  3;  i++ ) {
        ;
    } /*for*/

} /* end of function:  'wastetime()' */
```

Figure A-4. F_WASTE.LST

Now that we've run an assembly language program and viewed the trace results, let's explore the emulator's HLL (High Level Language) capabilities using the program F_DEMO.AOM. This program has been compiled and linked using Franklin's C-51 compiler and linker.

First, let's load the new file using the *File|Load* command. Enter **F_DEMO.AOM** and reply **N** to the merge prompt. This will clear the current environment set during the DEMO.DBG session.

Setting Breakpoints Revisited

Let's set another simple breakpoint. Select the *Break/Trace|Set|Add|CBREAK: Code Break-Point* command. Enter **0** for the start address and simply press **Enter** at the end address prompt.

Select the *CBREAK: Code Break-Point* command again. This time however, enter the start address using a source line number (see Appendix E, Using Symbols) as follows:

F_HLMAIN:#35

and again simply press **Enter** at the end address prompt.

Return to the Main Menu by pressing **Esc** and then select the *Run|Reset|Emulator* command to begin emulation. The breakpoint at location 0 will be encountered at once, as it was during the DEMO.DBG session. Select the *Run|Go* command to resume execution from the current PC. In a few moments, the breakpoint at location of the line number "F_HLMAIN:#35" is reached. Note that the source level statements will be present in the Source Window (we broke on line number 35 in F_HLMAIN so that line is highlighted as the next instruction to be executed). Now select the *Run|Go* command sequence once more. Now let's have a look at the Trace Buffer (Model 400 emulators only) again.

Viewing The Trace Buffer Revisited (Model 400 Emulators Only)

Select the *Break/Trace|View Trace* command. The address at line 35 of HLMAIN (the breakpoint) is at the trace trigger point (END). Select the *Search* command to search for an address and enter **F_HLMAIN:#35** at the address prompt. The first occurrence will be displayed. Repeat the *Search* and you will be moved back to the previous occurrence. You may have noticed that the previous entry in the search box was remembered the second time you used it. This is a convenient way to "search again" for another occurrence of the same item. Now press **Esc** to return to the Main Menu.

Complex Breakpoints

The Break Menu (*Break/Trace|Set*) allows complex breakpoints to be created. The method is much the same as using simple breakpoints, but you may AND together code address, direct addresses, bit addresses, opcode values, opcode class (see *Break/Trace|Set|Opcode Class* on page 7-61), and immediate operands.

First let's clear all current breakpoints by selecting the *Break/Trace|Clear* command (page 7-63). Now return to the Break Menu (*Break/Trace|Set*) and press **Tab** to enter the Complex Break Window. Let's set a complex breakpoint to halt execution on any 'MOV A,Daddr' instruction located within the subroutine WASTETIME. In order to do this, we can specify the start address and end address using line numbers.

To determine which line numbers span this routine we can use the *Source/Symbols|Line Number* window (page 7-54). From this display we can see the the function WASTETIME starts at F_WASTE:#10 and ends at F_WASTE:#14.

Now we can add a complex break. Select the *Break/Trace|Set|Add|CBREAK: Code Break-Point* command. In the Complex Break Window this automatically calls the *Break/Trace|Set|Edit* Pull-down Menu (page 7-59) from which you can choose which part of the complex break to enter (such as a code address range, direct address range, etc.). Select *Code Address Range* and enter **F_WASTE:#10** for the start address and **F_WASTE:#14** for the end address.

Return to the *Edit* Pull-down Menu and select *Opcode Value* and enter the value 0E5 (a hexadecimal number) which is the opcode for the instruction we are interested in. Now press **Esc** twice to return to the Main Menu. Note that the conditions you have defined will be evaluated to achieve the following condition:

if ((PC > = F_WASTE:#10 AND PC < = F_WASTE:#14) AND opcode = 0E5H) then Break

Now select the *Run|Reset|Emulator* command to start emulation. Emulation will halt at the first occurrence of the above condition. Note that breakpoints may be saved to a file using the *Break/Trace|Set|Save* command (page 7-63) and recalled later using the *Break/Trace|Set|Load* command (page 7-63).

Conclusion

This concludes the tutorial for the iceMASTER emulator. Please use the **Help** command in each menu to learn about and experiment with the capabilities and power of the iceMASTER emulator. Help is accessed by highlighting the command of interest and pressing Hot Key F1. Once a screen is displayed, highlighted topics can be hyperlinked or accessed in a read-more-about-it manner. To select a Hyperlink topic, press the **Tab** key to highlight the desired subject and press **Enter**. **Esc** will reverse the procedure.

Appendix B: Predefined Byte And Bit Addresses

The following tables detail the predefined byte and bit addresses for the 8051 family of microcontrollers supported by the MetaLink family of emulators. Proliferation parts are delimited from the standard MCS-51 definitions by boxes.

Predefined Byte Addresses

```
P0        DATA      080H      ;PORT 0
SP        DATA      081H      ;STACK POINTER
DPL       DATA      082H      ;DATA POINTER - LOW BYTE
DPH       DATA      083H      ;DATA POINTER - HIGH BYTE

---------- 80C321/80C521 -----------------------------------
DPL1      DATA      084H      ;DATA POINTER LOW 1
DPH1      DATA      085H      ;DATA POINTER HIGH 1
DPS       DATA      086H      ;DATA POINTER SELECTION

---------- 83C152/80C152 -----------------------------------
GMOD      DATA      084H      ;GSC MODE
TFIFO     DATA      085H      ;GSC TRANSMIT BUFFER

---------- 80C517/80C537 -----------------------------------
WDTREL    DATA      086H      ;WATCHDOG TIMER RELOAD REG

PCON      DATA      087H      ;POWER CONTROL
TCON      DATA      088H      ;TIMER CONTROL
TMOD      DATA      089H      ;TIMER MODE
TL0       DATA      08AH      ;TIMER 0 - LOW BYTE
TL1       DATA      08BH      ;TIMER 1 - LOW BYTE

---------- 83C751/83C752 -----------------------------------
RTL       DATA      08BH      ;TIMER 0 - LOW BYTE RELOAD

TH0       DATA      08CH      ;TIMER 0  HIGH BYTE
TH1       DATA      08DH      ;TIMER 1 - HIGH BYTE

---------- 83C751/83C752 -----------------------------------
RTH       DATA      08DH      ;TIMER 0 - HIGH BYTE RELOAD

---------- 83C752 ------------------------------------------
PWM       DATA      08EH      ;PULSE WIDTH MODULATION

P1        DATA      090H      ;PORT 1
```

```
-------- 83C152/80C152 --------
P5      DATA    091H    ;PORT 5
DCON0   DATA    092H    ;DMA CONTROL 0
DCON1   DATA    093H    ;DMA CONTROL 1
BAUD    DATA    094H    ;GSC BAUD RATE
ADR0    DATA    095H    ;GSC MATCH ADDRESS 0

-------- 80C452/83C452 --------
DCON0   DATA    092H    ;DMA CONTROL 0
DCON1   DATA    093H    ;DMA CONTROL 1

-------- 80C517/80C537 --------
DPSEL   DATA    092H    ;DATA POINTER SELECT REGISTER

SCON    DATA    098H    ;SERIAL PORT CONTROL
SBUF    DATA    099H    ;SERIAL PORT BUFFER

-------- 83C751/83C752 --------
I2CON   DATA    098H    ;I2C CONTROL
I2DAT   DATA    099H    ;I2C DATA

-------- 80C517/80C537 --------
IEN2    DATA    09AH    ;INTERRUPT ENABLE REGISTER 2
S1CON   DATA    09BH    ;SERIAL PORT CONTROL 1
S1BUF   DATA    09CH    ;SERIAL PORT BUFFER 1
S1REL   DATA    09DH    ;SERIAL RELOAD REG 1

-------- 83C053 --------
OSAT    DATA    098H    ;ON SCREEN ATTRIBUTES
OSDT    DATA    099H    ;ON SCREEN DATA
OSAD    DATA    09AH    ;ON SCREEN ADDRESS

P2      DATA    0A0H    ;PORT 2
IE      DATA    0A8H    ;INTERRUPT ENABLE

-------- 80C51FA/83C51FA(83C252/80C252) --------
SADDR   DATA    0A9H    ;SLAVE INDIVIDUAL ADDRESS

-------- 80515/80535 80C517/80C537 --------
IP0     DATA    0A9H    ;INTERRUPT PRIORITY REGISTER 0

-------- 80C321/80C521 --------
WDS     DATA    0A9H    ;WATCHDOG SELECTION
WDK     DATA    0AAH    ;WATCHDOG KEY

-------- 83C152/80C152 --------
P6      DATA    0A1H    ;PORT 6
SARL0   DATA    0A2H    ;DMA SOURCE ADDR. 0 (LOW)
SARH0   DATA    0A3H    ;DMA SOURCE ADDR. 0 (HIGH)
IFS     DATA    0A4H    ;GSC INTERFRAME SPACING
ADR1    DATA    0A5H    ;GSC MATCH ADDRESS 1

-------- 80C452/83C452 --------
SARL0   DATA    0A2H    ;DMA SOURCE ADDR. 0 (LOW)
SARH0   DATA    0A3H    ;DMA SOURCE ADDR. 0 (HIGH)

-------- 80C552/83C552 --------
CML0    DATA    0A9H    ;COMPARE 0 - LOW BYTE
CML1    DATA    0AAH    ;COMPARE 1 - LOW BYTE
CML2    DATA    0ABH    ;COMPARE 2 - LOW BYTE
CTL0    DATA    0ACH    ;CAPTURE 0 - LOW BYTE
CTL1    DATA    0ADH    ;CAPTURE 1 - LOW BYTE
CTL2    DATA    0AEH    ;CAPTURE 2 - LOW BYTE
CTL3    DATA    0AFH    ;CAPTURE 3 - LOW BYTE

P3      DATA    0B0H    ;PORT 3
```

83C152/80C152

SARL1	DATA	0B2H	;DMA SOURCE ADDR. 1 (LOW)
SARH1	DATA	0B3H	;DMA SOURCE ADDR. 1 (HIGH)
SLOTTM	DATA	0B4H	;GSC SLOT TIME
ADR2	DATA	0B5H	;GSC MATCH ADDRESS 2

80C452/83C452

SARL1	DATA	0B2H	;DMA SOURCE ADDR. 1 (LOW)
SARH1	DATA	0B3H	;DMA SOURCE ADDR. 1 (HIGH)

IP	DATA	0B8H	;INTERRUPT PRIORITY

80C51FA/83C51FA(83C252/80C252)

SADEN	DATA	0B9H	;SLAVE ADDRESS ENABLE

80515/80535 80C517/80C537

IP1	DATA	0B9H	;INTERRUPT PRIORITY REGISTER 1
IRCON	DATA	0C0H	;INTERRUPT REQUEST CONTROL
CCEN	DATA	0C1H	;COMPARE/CAPTURE ENABLE
CCL1	DATA	0C2H	;COMPARE/CAPTURE REGISTER 1 - LOW BYTE
CCH1	DATA	0C3H	;COMPARE/CAPTURE REGISTER 1 - HIGH BYTE
CCL2	DATA	0C4H	;COMPARE/CAPTURE REGISTER 2 - LOW BYTE
CCH2	DATA	0C5H	;COMPARE/CAPTURE REGISTER 2 - HIGH BYTE
CCL3	DATA	0C6H	;COMPARE/CAPTURE REGISTER 3 - LOW BYTE
CCH3	DATA	0C7H	;COMPARE/CAPTURE REGISTER 3 - HIGH BYTE
T2CON	DATA	0C8H	;TIMER 2 CONTROL
CRCL	DATA	0CAH	;COMPARE/RELOAD/CAPTURE - LOW BYTE
CRCH	DATA	0CBH	;COMPARE/RELOAD/CAPTURE - HIGH BYTE
TL2	DATA	0CCH	;TIMER 2 - LOW BYTE
TH2	DATA	0CDH	;TIMER 2 - HIGH BYTE

80C517/80C537

CC4EN	DATA	0C9H	;COMPARE/CAPTURE 4 ENABLE
CCL4	DATA	0CEH	;COMPARE/CAPTURE REGISTER 4 - LOW BYTE
CCH4	DATA	0CFH	;COMPARE/CAPTURE REGISTER 4 - HIGH BYTE

8044/8344

STS	DATA	0C8H	;SIU STATUS REGISTER
SMD	DATA	0C9H	;SERIAL MODE
RCB	DATA	0CAH	;RECEIVE CONTROL BYTE
RBL	DATA	0CBH	;RECEIVE BUFFER LENGTH
RBS	DATA	0CCH	;RECEIVE BUFFER START
RFL	DATA	0CDH	;RECEIVE FIELD LENGTH
STAD	DATA	0CEH	;STATION ADDRESS
DMA_CNT	DATA	0CFH	;DMA COUNT

8052/8032, 80C51FA/83C51FA(83C252/80C252), 80C154/83C154

T2CON	DATA	0C8H	;TIMER 2 CONTROL

80C51FA/83C51FA(83C252/80C252)

T2MOD	DATA	0C9H	;TIMER 2 MODE CONTROL

8052/8032, 80C51FA/83C51FA(83C252/80C252), 80C154/83C154

RCAP2L	DATA	0CAH	;TIMER 2 CAPTURE REGISTER, LOW BYTE
RCAP2H	DATA	0CBH	;TIMER 2 CAPTURE REGISTER, HIGH BYTE
TL2	DATA	0CCH	;TIMER 2 LOW BYTE
TH2	DATA	0CDH	;TIMER 2 - HIGH BYTE

83C152/80C152

P4	DATA	0C0H	;PORT 4
DARL0	DATA	0C2H	;DMA DESTINATION ADDR. 0 (LOW)
DARH0	DATA	0C3H	;DMA DESTINATION ADDR. 0 (HIGH)
BKOFF	DATA	0C4H	;GSC BACKOFF TIMER
ADR3	DATA	0C5H	;GSC MATCH ADDRESS 3
IEN1	DATA	0C8H	;INTERRUPT ENABLE REGISTER 1

80C452/83C452

P4	DATA	0C0H	;PORT 4
DARL0	DATA	0C2H	;DMA DESTINATION ADDR. 0 (LOW)
DARH0	DATA	0C3H	;DMA DESTINATION ADDR. 0 (HIGH)

80C451/83C451

```
P4      DATA    0C0H    ;PORT 4
P5      DATA    0C8H    ;PORT 5
```

80512/80532

```
IRCON   DATA    0C0H    ;INTERRUPT REQUEST CONTROL
```

80C552/83C552

```
P4      DATA    0C0H    ;PORT 4
P5      DATA    0C4H    ;PORT 5
ADCON   DATA    0C5H    ;A/D CONVERTER CONTROL
ADCH    DATA    0C6H    ;A/D CONVERTER HIGH BYTE
TM2IR   DATA    0C8H    ;T2 INTERRUPT FLAGS
CMH0    DATA    0C9H    ;COMPARE 0 - HIGH BYTE
CMH1    DATA    0CAH    ;COMPARE 1 - HIGH BYTE
CMH2    DATA    0CBH    ;COMPARE 2 - HIGH BYTE
CTH0    DATA    0CCH    ;CAPTURE 0 - HIGH BYTE
CTH1    DATA    0CDH    ;CAPTURE 1 - HIGH BYTE
CTH2    DATA    0CEH    ;CAPTURE 2 - HIGH BYTE
CTH3    DATA    0CFH    ;CAPTURE 3 - HIGH BYTE
```

83C053

```
OSCON   DATA    0C0H    ;ON SCREEN DISPLAY CONTROL
OSMOD   DATA    0C1H    ;ON SCREEN DISPLAY MODE
OSORG   DATA    0C2H    ;ON SCREEN DISPLAY ORIGIN
```

```
PSW     DATA    0D0H    ;PROGRAM STATUS WORD
```

8044/8344

```
NSNR    DATA    0D8H    ;SEND COUNT/RECEIVE COUNT
SIUST   DATA    0D9H    ;SIU STATE COUNTER
TCB     DATA    0DAH    ;TRANSMIT CONTROL BYTE
TBL     DATA    0DBH    ;TRANSMIT BUFFER LENGTH
TBS     DATA    0DCH    ;TRANSMIT BUFFER START
FIFO0   DATA    0DDH    ;THREE BYTE FIFO
FIFO1   DATA    0DEH
FIFO2   DATA    0DFH
```

80C51FA/83C51FA(83C252/80C252)

```
CCON    DATA    0D8H    ;CONTROL COUNTER
CMOD    DATA    0D9H    ;COUNTER MODE
CCAPM0  DATA    0DAH    ;COMPARE/CAPTURE MODE FOR PCA MODULE 0
CCAPM1  DATA    0DBH    ;COMPARE/CAPTURE MODE FOR PCA MODULE 1
CCAPM2  DATA    0DCH    ;COMPARE/CAPTURE MODE FOR PCA MODULE 2
CCAPM3  DATA    0DDH    ;COMPARE/CAPTURE MODE FOR PCA MODULE 3
CCAPM4  DATA    0DEH    ;COMPARE/CAPTURE MODE FOR PCA MODULE 4
```

80515/80535

```
ADCON   DATA    0D8H    ;A/D CONVERTER CONTROL
ADDAT   DATA    0D9H    ;A/D CONVERTER DATA
DAPR    DATA    0DAH    ;D/A CONVERTER PROGRAM REGISTER
```

83C152/80C152

```
DARL1   DATA    0D2H    ;DMA DESTINATION ADDR. 1 (LOW)
DARH1   DATA    0D3H    ;DMA DESTINATION ADDR. 1 (HIGH)
TCDCNT  DATA    0D4H    ;GSC TRANSMIT COLLISION COUNTER
AMSK0   DATA    0D5H    ;GSC ADDRESS MASK 0
TSTAT   DATA    0D8H    ;TRANSMIT STATUS (DMA & GSC)
```

80C452/83C452

```
DARL1   DATA    0D2H    ;DMA DESTINATION ADDR. 1 (LOW)
DARH1   DATA    0D3H    ;DMA DESTINATION ADDR. 1 (HIGH)
```

80C451/83C451

```
P6      DATA    0D8H    ;PORT 6
```

80512/80532

```
ADCON   DATA    0D8H    ;A/D CONVERTER CONTROL
ADDAT   DATA    0D9H    ;A/D CONVERTER DATA
DAPR    DATA    0DAH    ;D/A CONVERTER PROGRAM REGISTER
P6      DATA    0DBH    ;PORT 6
```

```
------ 83C751/83C752 ------
I2CFG    DATA    0D8H    ;I2C CONFIGURATION

------ 80C552/83C552, 80C652/83C652 ------
S1CON    DATA    0D8H    ;SERIAL 1 CONTROL
S1STA    DATA    0D9H    ;SERIAL 1 STATUS
S1DAT    DATA    0DAH    ;SERIAL 1 DATA
S1ADR    DATA    0DBH    ;SERIAL 1 SLAVE ADDRESS

------ 80C517/80C537 ------
CML0     DATA    0D2H    ;COMPARE REGISTER 0  - LOW BYTE
CMH0     DATA    0D3H    ;COMPARE REGISTER 0  - HIGH BYTE
CML1     DATA    0D4H    ;COMPARE REGISTER 1  - LOW BYTE
CMH1     DATA    0D5H    ;COMPARE REGISTER 1  - HIGH BYTE
CML2     DATA    0D6H    ;COMPARE REGISTER 2  - LOW BYTE
CMH2     DATA    0D7H    ;COMPARE REGISTER 2  - HIGH BYTE
ADCON0   DATA    0D8H    ;A/D CONVERTER CONTROL 0
ADDAT    DATA    0D9H    ;A/D CONVERTER DATA
DAPR     DATA    0DAH    ;D/A CONVERTER PROGRAM REGISTER
P7       DATA    0DBH    ;PORT 7
ADCON1   DATA    0DCH    ;A/D CONVERTER CONTROL 1
P8       DATA    0DDH    ;PORT 8
CTRELL   DATA    0DEH    ;COM TIMER REL REG - LOW BYTE
CTRELH   DATA    0DFH    ;COM TIMER REL REG - HIGH BYTE

------ 83C053 ------
TDACL    DATA    0D2H    ;HI-RES PULSE WIDTH MODULATOR
TDACH    DATA    0D3H    ;HI-RES PULSE WIDTH MODULATOR
PWM0     DATA    0D4H    ;LO-RES PULSE WIDTH MODULATOR 0
PWM1     DATA    0D5H    ;LO-RES PULSE WIDTH MODULATOR 1
PWM2     DATA    0D6H    ;LO-RES PULSE WIDTH MODULATOR 2
PWM3     DATA    0D7H    ;LO-RES PULSE WIDTH MODULATOR 3
SAD      DATA    0D8H    ;D/A AND VOLTAGE COMPARATOR
PWM4     DATA    0DCH    ;LO-RES PULSE WIDTH MODULATOR 4
PWM5     DATA    0DDH    ;LO-RES PULSE WIDTH MODULATOR 5
PWM6     DATA    0DEH    ;LO-RES PULSE WIDTH MODULATOR 6
PWM7     DATA    0DFH    ;LO-RES PULSE WIDTH MODULATOR 7

ACC      DATA    0E0H    ;ACCUMULATOR

------ 83C152/80C152 ------
BCRL0    DATA    0E2H    ;DMA BYTE COUNT 0 (LOW)
BCRH0    DATA    0E3H    ;DMA BYTE COUNT 0 (HIGH)
PRBS     DATA    0E4H    ;GSC PSEUDO-RANDOM SEQUENCE
AMSK1    DATA    0E5H    ;GSC ADDRESS MASK 1
RSTAT    DATA    0E8H    ;RECEIVE STATUS (DMA & GSC)

------ 80C452/83C452 ------
BCRL0    DATA    0E2H    ;DMA BYTE COUNT 0 (LOW)
BCRH0    DATA    0E3H    ;DMA BYTE COUNT 0 (HIGH)
HSTAT    DATA    0E6H    ;HOST STATUS
HCON     DATA    0E7H    ;HOST CONTROL
SLCON    DATA    0E8H    ;SLAVE CONTROL
SSTAT    DATA    0E9H    ;SLAVE STATUS
IWPR     DATA    0EAH    ;INPUT WRITE POINTER
IRPR     DATA    0EBH    ;INPUT READ POINTER
CBP      DATA    0ECH    ;CHANNEL BOUNDARY POINTER
FIN      DATA    0EEH    ;FIFO IN
CIN      DATA    0EFH    ;COMMAND IN

------ 80515/80535 ------
P4       DATA    0E8H    ;PORT 4

------ 80C451/83C451 ------
CSR      DATA    0E8H    ;CONTROL STATUS

------ 80512/80532 ------
P4       DATA    0E8H    ;PORT 4
```

80C552/83C552

```
IEN1      DATA    0E8H    ;INTERRUPT ENABLE REGISTER 1
TM2CON    DATA    0EAH    ;T2 COUNTER CONTROL
CTCON     DATA    0EBH    ;CAPTURE CONTROL
TML2      DATA    0ECH    ;TIMER 2 - LOW BYTE
TMH2      DATA    0EDH    ;TIMER 2 - HIGH BYTE
STE       DATA    0EEH    ;SET ENABLE
RTE       DATA    0EFH    ;RESET/TOGGLE ENABLE
```

80C51FA/83C51FA(83C252/80C262)

```
CL        DATA    0E9H    ;CAPTURE BYTE LOW
CCAP0L    DATA    0EAH    ;COMPARE/CAPTURE 0 LOW BYTE
CCAP1L    DATA    0EBH    ;COMPARE/CAPTURE 1 LOW BYTE
CCAP2L    DATA    0ECH    ;COMPARE/CAPTURE 2 LOW BYTE
CCAP3L    DATA    0EDH    ;COMPARE/CAPTURE 3 LOW BYTE
CCAP4L    DATA    0EEH    ;COMPARE/CAPTURE 4 LOW BYTE
```

80C517/80C537

```
CTCON     DATA    0E1H    ;COM TIMER CONTROL REG
CML3      DATA    0E2H    ;COMPARE REGISTER 3  - LOW BYTE
CMH3      DATA    0E3H    ;COMPARE REGISTER 3  - HIGH BYTE
CML4      DATA    0E4H    ;COMPARE REGISTER 4  - LOW BYTE
CMH4      DATA    0E5H    ;COMPARE REGISTER 4  - HIGH BYTE
CML5      DATA    0E6H    ;COMPARE REGISTER 5  - LOW BYTE
CMH5      DATA    0E7H    ;COMPARE REGISTER 5  - HIGH BYTE
P4        DATA    0E8H    ;PORT 4
MD0       DATA    0E9H    ;MUL/DIV REG 0
MD1       DATA    0EAH    ;MUL/DIV REG 1
MD2       DATA    0EBH    ;MUL/DIV REG 2
MD3       DATA    0ECH    ;MUL/DIV REG 3
MD4       DATA    0EDH    ;MUL/DIV REG 4
MD5       DATA    0EEH    ;MUL/DIV REG 5
ARCON     DATA    0EFH    ;ARITHMETIC CONTROL REG
```

```
B         DATA    0F0H    ;MULTIPLICATION REGISTER
```

80C154/83C154

```
IOCON     DATA    0F8H    ;I/O CONTROL REGISTER
```

83C152/80C152

```
BCRL1     DATA    0F2H    ;DMA BYTE COUNT 1 (LOW)
BCRH1     DATA    0F3H    ;DMA BYTE COUNT 1 (HIGH)
RFIFO     DATA    0F4H    ;GSC RECEIVE BUFFER
MYSLOT    DATA    0F5H    ;GSC SLOT ADDRESS
IPN1      DATA    0F8H    ;INTERRUPT PRIORITY REGISTER 1
```

80C452/83C452

```
BCRL1     DATA    0F2H    ;DMA BYTE COUNT 1 (LOW)
BCRH1     DATA    0F3H    ;DMA BYTE COUNT 1 (HIGH)
ITHR      DATA    0F6H    ;INPUT FIFO THRESHOLD
OTHR      DATA    0F7H    ;OUTPUT FIFO THRESHOLD
IEP       DATA    0F8H    ;INTERRUPT PRIORITY
MODE      DATA    0F9H    ;MODE
ORPR      DATA    0FAH    ;OUTPUT READ POINTER
OWPR      DATA    0FBH    ;OUTPUT WRITE POINTER
IMIN      DATA    0FCH    ;IMMEDIATE COMMAND IN
IMOUT     DATA    0FDH    ;IMMEDIATE COMMAND OUT
FOUT      DATA    0FEH    ;FIFO OUT
COUT      DATA    0FFH    ;COMMAND OUT
```

80515/80535

```
P5        DATA    0F8H    ;PORT 5
```

80512/80532

```
P5        DATA    0F8H    ;PORT 5
```

83C751/83C752

```
I2STA     DATA    0F8H    ;I2C STATUS
```

80C552/83C552

```
IP1      DATA    0F8H    ;INTERRUPT PRIORITY REGISTER 1
PWM0     DATA    0FCH    ;PULSE WIDTH REGISTER 0
PWM1     DATA    0FDH    ;PULSE WIDTH REGISTER 1
PWMP     DATA    0FEH    ;PRESCALER FREQUENCY CONTROL
T3       DATA    0FFH    ;T3 - WATCHDOG TIMER
```

80C517/80C537

```
CML6     DATA    0F2H    ;COMPARE REGISTER 6  - LOW BYTE
CMH6     DATA    0F3H    ;COMPARE REGISTER 6  - HIGH BYTE
CML7     DATA    0F4H    ;COMPARE REGISTER 7  - LOW BYTE
CMH7     DATA    0F5H    ;COMPARE REGISTER 7  - HIGH BYTE
CMEN     DATA    0F6H    ;COMPARE ENABLE
CMSEL    DATA    0F7H    ;COMPARE INPUT REGISTER
P5       DATA    0F8H    ;PORT 5
P6       DATA    0FAH    ;PORT 6
```

80C51FA/83C51FA(83C252/80C252)

```
CH       DATA    0F9H    ;CAPTURE HIGH BYTE
CCAP0H   DATA    0FAH    ;COMPARE/CAPTURE 0 HIGH BYTE
CCAP1H   DATA    0FBH    ;COMPARE/CAPTURE 1 HIGH BYTE
CCAP2H   DATA    0FCH    ;COMPARE/CAPTURE 2 HIGH BYTE
CCAP3H   DATA    0FDH    ;COMPARE/CAPTURE 3 HIGH BYTE
CCAP4H   DATA    0FEH    ;COMPARE/CAPTURE 4 HIGH BYTE
```

83C752

```
PWENA    DATA    0FEH    ;PULSE WIDTH ENABLE
```

80C851/83C851

```
EADRL    DATA    0F2H    ;EEPROM ADDRESS REGISTER - LOW BYTE
EADRH    DATA    0F3H    ;EEPROM ADDRESS REGISTER - HIGH BYTE
EDAT     DATA    0F4H    ;EEPROM DATA REGISTER
ETIM     DATA    0F5H    ;EEPROM TIMER REGISTER
ECNTRL   DATA    0F6H    ;EEPROM CONTROL REGISTER
```

Predefined Bit Addresses

83C751/83C752

SCL	BIT	080H	;P0.0 - I2C SERIAL CLOCK
SDA	BIT	081H	;P0.1 - I2C SERIAL DATA

IT0	BIT	088H	;TCON.0 - EXT. INTERRUPT 0 TYPE
IE0	BIT	089H	;TCON.1 - EXT. INTERRUPT 0 EDGE FLAG
IT1	BIT	08AH	;TCON.2 - EXT. INTERRUPT 1 TYPE
IE1	BIT	08BH	;TCON.3 - EXT. INTERRUPT 1 EDGE FLAG
TR0	BIT	08CH	;TCON.4 - TIMER 0 ON/OFF CONTROL
TF0	BIT	08DH	;TCON.5 - TIMER 0 OVERFLOW FLAG
TR1	BIT	08EH	;TCON.6 - TIMER 1 ON/OFF CONTROL
TF1	BIT	08FH	;TCON.7 - TIMER 1 OVERFLOW FLAG

83C751/83C752

C/T	BIT	08EH	;TCON.6 - COUNTER OR TIMER OPERATION
GATE	BIT	08FH	;TCON.7 - GATE TIMER

80515/80535

INT3	BIT	090H	;P1.0 - EXT. INTERRUPT 3/CAPT & COMP 0
INT4	BIT	091H	;P1.1 - EXT. INTERRUPT 4/CAPT & COMP 1
INT5	BIT	092H	;P1.2 - EXT. INTERRUPT 5/CAPT & COMP 2
INT6	BIT	093H	;P1.3 - EXT. INTERRUPT 6/CAPT & COMP 3
INT2	BIT	094H	;P1.4 - EXT. INTERRUPT 2
T2EX	BIT	095H	;P1.5 - TIMER 2 EXT. RELOAD TRIGGER INP
CLKOUT	BIT	096H	;P1.6 - SYSTEM CLOCK OUTPUT
T2	BIT	097H	;P1.7 - TIMER 2 INPUT

83C152/80C152

GRXD	BIT	090H	;P1.0 - GSC RECEIVER DATA INPUT
GTXD	BIT	091H	;P1.1 - GSC TRANSMITTER DATA OUTPUT
DEN	BIT	092H	;P1.2 - DRIVE ENABLE TO ENABLE EXT DRIVER
TXC	BIT	093H	;P1.3 - GSC EXTERNAL TRANSMIT CLOCK INPUT
RXC	BIT	094H	;P1.4 - GSC EXTERNAL RECEIVER CLOCK INPUT

83C552/80C552

CT0I	BIT	090H	;P1.0 - CAPTURE/TIMER INPUT 0
CT1I	BIT	091H	;P1.1 - CAPTURE/TIMER INPUT 1
CT2I	BIT	092H	;P1.2 - CAPTURE/TIMER INPUT 2
CT3I	BIT	093H	;P1.3 - CAPTURE/TIMER INPUT 3
T2	BIT	094H	;P1.4 - T2 EVENT INPUT
RT2	BIT	095H	;P1.5 - T2 TIMER RESET SIGNAL
SCL	BIT	096H	;P1.6 - SERIAL PORT CLOCK LINE I2C
SDA	BIT	097H	;P1.7 - SERIAL PORT DATA LINE I2C

80C517/80C537

INT3	BIT	090H	;P1.0 - EXT. INTERRUPT 3/CAPT & COMP 0
INT4	BIT	091H	;P1.1 - EXT. INTERRUPT 4/CAPT & COMP 1
INT5	BIT	092H	;P1.2 - EXT. INTERRUPT 5/CAPT & COMP 2
INT6	BIT	093H	;P1.3 - EXT. INTERRUPT 6/CAPT & COMP 3
INT2	BIT	094H	;P1.4 - EXT. INTERRUPT 2
T2EX	BIT	095H	;P1.5 - TIMER 2 EXT. RELOAD TRIGGER INPUT
CLKOUT	BIT	096H	;P1.6 - SYSTEM CLOCK OUTPUT
T2	BIT	097H	;P1.7 - TIMER 2 INPUT

80C452/83C452, 80C152/83C152

HLD	BIT	095H	;P1.5 - DMA HOLD REQUEST I/O
HLDA	BIT	096H	;P1.6 - DMA HOLD ACKNOWLEDGE OUTPUT

83C751/83C752

INT0	BIT	095H	;P1.5 - EXTERNAL INTERRUPT 0 INPUT
INT1	BIT	096H	;P1.6 - EXTERNAL INTERRUPT 1 INPUT
T0	BIT	096H	;P1.7 - TIMER 0 COUNT INPUT

RI	BIT	098H	;SCON.0 - RECEIVE INTERRUPT FLAG
TI	BIT	099H	;SCON.1 - TRANSMIT INTERRUPT FLAG
RB8	BIT	09AH	;SCON.2 - RECEIVE BIT 8
TB8	BIT	09BH	;SCON.3 - TRANSMIT BIT 8
REN	BIT	09CH	;SCON.4 - RECEIVE ENABLE
SM2	BIT	09DH	;SCON.5 - SERIAL MODE CONTROL BIT 2
SM1	BIT	09EH	;SCON.6 - SERIAL MODE CONTROL BIT 1

```
SMO     BIT         09FH    ;SCON.7 - SERIAL MODE CONTROL BIT 0

------- 83C751/83C752 -------
MASTER  BIT(READ) 099H      ;I2CON.1 - MASTER
STP     BIT(READ) 09AH      ;I2CON.2 - STOP
STR     BIT(READ) 09BH      ;I2CON.3 - START
ARL     BIT(READ) 09CH      ;I2CON.4 - ARBITRATION LOSS
DRDY    BIT(READ) 09DH      ;I2CON.5 - DATA READY
ATN     BIT(READ) 09EH      ;I2CON.6 - ATTENTION
RDAT    BIT(READ) 09FH      ;I2CON.7 - RECEIVE DATA
XSTP    BIT(WRITE)098H      ;I2CON.0 - TRANSMIT STOP
XSTR    BIT(WRITE)099H      ;I2CON.1 - TRANSMIT REPEATED START
CSTP    BIT(WRITE)09AH      ;I2CON.2 - CLEAR STOP
CSTR    BIT(WRITE)09BH      ;I2CON.3 - CLEAR START
CARL    BIT(WRITE)09CH      ;I2CON.4 - CLEAR ARBITRATION LOSS
CDR     BIT(WRITE)09DH      ;I2CON.5 - CLEAR DATA READY
IDLE    BIT(WRITE)09EH      ;I2CON.6 - GO IDLE
CXA     BIT(WRITE)09FH      ;I2CON.7 - CLEAR TRANSMIT ACTIVE

------- 83C053 -------
ADI0    BIT         090H    ;P1.0 - SOFTWARE A/D INPUT 0
ADI1    BIT         091H    ;P1.1 - SOFTWARE A/D INPUT 1
ADI2    BIT         092H    ;P1.2 - SOFTWARE A/D INPUT 2
ASAT0   BIT         098H    ;OSAT.0 - MULTIPLE FUNCTION (DEPENDS ON OSDT)
ASAT1   BIT         099H    ;OSAT.1 - MULTIPLE FUNCTION (DEPENDS ON OSDT)
ASAT2   BIT         09AH    ;OSAT.2 - MULTIPLE FUNCTION (DEPENDS ON OSDT)
ASAT3   BIT         09BH    ;OSAT.3 -
ASAT4   BIT         09CH    ;OSAT.4 - SHOW BACKGROUND / SHOW FURTHER ROWS
ASAT5   BIT         09DH    ;OSAT.5 -
ASAT6   BIT         09EH    ;OSAT.6 -
ASAT7   BIT         09FH    ;OSAT.7 -

EX0     BIT         0A8H    ;IE.0 - EXTERNAL INTERRUPT 0 ENABLE
ET0     BIT         0A9H    ;IE.1 - TIMER 0 INTERRUPT ENABLE
EX1     BIT         0AAH    ;IE.2 - EXTERNAL INTERRUPT 1 ENABLE
ET1     BIT         0ABH    ;IE.3 - TIMER 1 INTERRUPT ENABLE
ES      BIT         0ACH    ;IE.4 - SERIAL PORT INTERRUPT ENABLE

------- 83C751/83C752 -------
EI2     BIT         0ACH    ;IE.4 - SERIAL PORT INTERRUPT ENABLE

------- 83C053 -------
EVS     BIT         0ACH    ;IE.4 - VSYNC INTERRUPT ENABLE

------- 8052/8032, 80C154/83C154, 80C252(80C51FA), 80515/80535 -------
ET2     BIT         0ADH    ;TIMER 2 INTERRUPT ENABLE

------- 80C652/83C652 -------
ES1     BIT         0ADH    ;IE.5 - SERIAL PORT 1 INTERRUPT ENABLE

------- 80C252(80C51FA) -------
EC      BIT         0AEH    ;IE.6 - ENABLE PCA INTERRUPT

------- 80515/80535 -------
WDT     BIT         0AEH    ;IEN0.6 - WATCHDOG TIMER RESET

------- 83C552/80C552 -------
ES1     BIT         0ADH    ;IEN0.5 - SERIAL PORT 1 INTERRUPT ENABLE
EAD     BIT         0AEH    ;IEN0.6 - ENABLE A/D INTERRUPT

------- 80C517/80C537 -------
ET2     BIT         0ADH    ;IEN0.5 - TIMER 2 INTERRUPT ENABLE
WDT     BIT         0AEH    ;IEN0.6 - WATCHDOG TIMER RESET

EA      BIT         0AFH    ;IE.7 - GLOBAL INTERRUPT ENABLE
RXD     BIT         0B0H    ;P3.0 - SERIAL PORT RECEIVE INPUT
TXD     BIT         0B1H    ;P3.1 - SERIAL PORT TRANSMIT OUTPUT
INT0    BIT         0B2H    ;P3.2 - EXTERNAL INTERRUPT 0 INPUT
INT1    BIT         0B3H    ;P3.3 - EXTERNAL INTERRUPT 1 INPUT
T0      BIT         0B4H    ;P3.4 - TIMER 0 COUNT INPUT
T1      BIT         0B5H    ;P3.5 - TIMER 1 COUNT INPUT
WR      BIT         0B6H    ;P3.6 - WRITE CONTROL FOR EXT. MEMORY
RD      BIT         0B7H    ;P3.7 - READ CONTROL FOR EXT. MEMORY
```

```
PX0     BIT     0B8H    ;IP.0 - EXTERNAL INTERRUPT 0 PRIORITY
PT0     BIT     0B9H    ;IP.1 - TIMER 0 PRIORITY
PX1     BIT     0BAH    ;IP.2 - EXTERNAL INTERRUPT 1 PRIORITY
PT1     BIT     0BBH    ;IP.3 - TIMER 1 PRIORITY
PS      BIT     0BCH    ;IP.4 - SERIAL PORT PRIORITY

        80C154/83C154
PT2     BIT     0BCH    ;IP.5 - TIMER 2 PRIORITY
PCT     BIT     0BFH    ;IP.7 - INTERRUPT PRIORITY DISABLE

        80C652/83C652
PS1     BIT     0BDH    ;IP.5 - SERIAL PORT 1 PRIORITY

        80C51FA/83C51FA(83C252/80C252)
PT2     BIT     0BDH    ;IP.5 - TIMER 2 PRIORITY
PPC     BIT     0BEH    ;IP.6 - PCA PRIORITY

        80515/80535, 80C517/80C537
EADC    BIT     0B8H    ;IEN1.0 - A/D CONVERTER INTERRUPT ENABLE
EX2     BIT     0B9H    ;IEN1.1 - EXT. INTERRUPT 2 ENABLE
EX3     BIT     0BAH    ;IEN1.2 - EXT. INT 3/CAPT/COMP INT 0 EN
EX4     BIT     0BBH    ;IEN1.3 - EXT. INT 4/CAPT/COMP INT 1 EN
EX5     BIT     0BCH    ;IEN1.4 - EXT. INT 5/CAPT/COMP INT 2 EN
EX6     BIT     0BDH    ;IEN1.5 - EXT. INT 6/CAPT/COMP INT 3 EN
SWDT    BIT     0BEH    ;IEN1.6 - WATCHDOG TIMER START
EXEN2   BIT     0BFH    ;IEN1.7 - T2 EXT. RELOAD INTER START
IADC    BIT     0C0H    ;IRCON.0 - A/D CONVERTER INTER REQUEST
IEX2    BIT     0C1H    ;IRCON.1 - EXT. INTERRUPT 2 EDGE FLAG
IEX3    BIT     0C2H    ;IRCON.2 - EXT. INTERRUPT 3 EDGE FLAG
IEX4    BIT     0C3H    ;IRCON.3 - EXT. INTERRUPT 4 EDGE FLAG
IEX5    BIT     0C4H    ;IRCON.4 - EXT. INTERRUPT 5 EDGE FLAG
IEX6    BIT     0C5H    ;IRCON.5 - EXT. INTERRUPT 6 EDGE FLAG
TF2     BIT     0C6H    ;IRCON.6 - TIMER 2 OVERFLOW FLAG
EXF2    BIT     0C7H    ;IRCON.7 - TIMER 2 EXT. RELOAD FLAG
T2I0    BIT     0C8H    ;T2CON.0 - TIMER 2 INPUT SELECT BIT 0
T2I1    BIT     0C9H    ;T2CON.1 - TIMER 2 INPUT SELECT BIT 1
T2CM    BIT     0CAH    ;T2CON.2 - COMPARE MODE
T2R0    BIT     0CBH    ;T2CON.3 - TIMER 2 RELOAD MODE SEL BIT 0
T2R1    BIT     0CCH    ;T2CON.4 - TIMER 2 RELOAD MODE SEL BIT 1
I2FR    BIT     0CDH    ;T2CON.5 - EXT. INT 2 F/R EDGE FLAG
I3FR    BIT     0CEH    ;T2CON.6 - EXT. INT 3 F/R EDGE FLAG
T2PS    BIT     0CFH    ;T2CON.7 - PRESCALER SELECT BIT

        80C552/83C552
PS1     BIT     0BDH    ;IP0.5 - SIO1
PAD     BIT     0BEH    ;IP0.6 - A/D CONVERTER
CMSR0   BIT     0C0H    ;P4.0 - T2 COMPARE AND SET/RESET OUTPUTS
CMSR1   BIT     0C1H    ;P4.1 - T2 COMPARE AND SET/RESET OUTPUTS
CMSR2   BIT     0C2H    ;P4.2 - T2 COMPARE AND SET/RESET OUTPUTS
CMSR3   BIT     0C3H    ;P4.3 - T2 COMPARE AND SET/RESET OUTPUTS
CMSR4   BIT     0C4H    ;P4.4 - T2 COMPARE AND SET/RESET OUTPUTS
CMSR5   BIT     0C5H    ;P4.5 - T2 COMPARE AND SET/RESET OUTPUTS
CMT0    BIT     0C6H    ;P4.6 - T2 COMPARE AND TOGGLE OUTPUTS
CMT1    BIT     0C7H    ;P4.7 - T2 COMPARE AND TOGGLE OUTPUTS
CTI0    BIT     0C8H    ;TM2IR.0 - T2 CAPTURE 0
CTI1    BIT     0C9H    ;TM2IR.1 - T2 CAPTURE 1
CTI2    BIT     0CAH    ;TM2IR.2 - T2 CAPTURE 2
CTI3    BIT     0CBH    ;TM2IR.3 - T2 CAPTURE 3
CMI0    BIT     0CCH    ;TM2IR.4 - T2 COMPARATOR 0
CMI1    BIT     0CDH    ;TM2IR.5 - T2 COMPARATOR 1
CMI2    BIT     0CEH    ;TM2IR.6 - T2 COMPARATOR 2
T2OV    BIT     0CFH    ;TM2IR.7 - T2 OVERFLOW

        8044/8344
RBP     BIT     0C8H    ;STS.0 - RECEIVE BUFFER PROTECT
AM      BIT     0C9H    ;STS.1 - AUTO/ADDRESSED MODE SELECT
OPB     BIT     0CAH    ;STS.2 - OPTIONAL POLL BIT
BOV     BIT     0CBH    ;STS.3 - RECEIVE BUFFER OVERRUN
SI      BIT     0CCH    ;STS.4 - SIU INTERRUPT FLAG
RTS     BIT     0CDH    ;STS.5 - REQUEST TO SEND
RBE     BIT     0CEH    ;STS.6 - RECEIVE BUFFER EMPTY
TBF     BIT     0CFH    ;STS.7 - TRANSMIT BUFFER FULL
```

8052/8032, 80C154/83C154, 80C51FA/83C51FA(80C252/83C252)

```
CAP2     BIT     0C8H     ;T2CON.0 - CAPTURE OR RELOAD SELECT
CNT2     BIT     0C9H     ;T2CON.1 - TIMER OR COUNTER SELECT
TR2      BIT     0CAH     ;T2CON.2 - TIMER 2 ON/OFF CONTROL
EXEN2    BIT     0CBH     ;T2CON.3 - TIMER 2 EXTERNAL ENABLE FLAG
TCLK     BIT     0CCH     ;T2CON.4 - TRANSMIT CLOCK SELECT
RCLK     BIT     0CDH     ;T2CON.5 - RECEIVE CLOCK SELECT
EXF2     BIT     0CEH     ;T2CON.6 - EXTERNAL TRANSITION FLAG
TF2      BIT     0CFH     ;T2CON.7 - TIMER 2 OVERFLOW FLAG
```

80C152/83C152

```
EGSRV    BIT     0C8H     ;IEN1.0 - GSC RECEIVE VALID
EGSRE    BIT     0C9H     ;IEN1.1 - GSC RECEIVE ERROR
EDMA0    BIT     0CAH     ;IEN1.2 - DMA CHANNEL REQUEST 0
EGSTV    BIT     0CBH     ;IEN1.3 - GSC TRANSMIT VALID
EDMA1    BIT     0CCH     ;IEN1.4 - DMA CHANNEL REQUEST 1
EGSTE    BIT     0CDH     ;IEN1.5 - GSC TRANSMIT ERROR
```

80512/80532

```
IADC     BIT     0C0H     ;IRCON.0 - A/D CONVERTER INTERRUPT REQ
```

83C053

```
BFE      BIT     0C0H     ;OSCON.0 - BF PIN ENABLE
DH       BIT     0C1H     ;OSCON.1 - DOUBLE HEIGHT
PO       BIT     0C2H     ;OSCON.2 - SHADOWING LOW/HIGH CONTROL
PC       BIT     0C3H     ;OSCON.3 - LOW / HIGH ON VCTRL (COLOR VID2:0)
PH       BIT     0C4H     ;OSCON.4 - HSYNC INPUT ACTIVE LOW / HIGH
LV       BIT     0C5H     ;OSCON.5 - VSYNC LEADING / TRAILING EDGE
PV       BIT     0C6H     ;OSCON.6 - VSYNC ACTIVE LOW / HIGH
IV       BIT     0C7H     ;OSCON.7 - VSYNC INTERRUPT FLAG
```

```
P        BIT     0D0H     ;PSW.0 - ACCUMULATOR PARITY FLAG
```

80C552/83C552

```
F1       BIT     0D1H     ;PSW.1 - FLAG
```

80512/80532

```
F1       BIT     0D1H     ;PSW.1 - FLAG
MX0      BIT     0D8H     ;ADCON.0 - ANALOG INPUT CH SELECT BIT 0
MX1      BIT     0D9H     ;ADCON.1 - ANALOG INPUT CH SELECT BIT 1
MX2      BIT     0DAH     ;ADCON.2 - ANALOG INPUT CH SELECT BIT 2
ADM      BIT     0DBH     ;ADCON.3 - A/D CONVERSION MODE
BSY      BIT     0DCH     ;ADCON.4 - BUSY FLAG
BD       BIT     0DFH     ;ADCON.7 - BAUD RATE ENABLE
```

```
OV       BIT     0D2H     ;PSW.2 - OVERFLOW FLAG
RS0      BIT     0D3H     ;PSW.3 - REGISTER BANK SELECT 0
RS1      BIT     0D4H     ;PSW.4 - REGISTER BANK SELECT 1
F0       BIT     0D5H     ;PSW.5 - FLAG 0
AC       BIT     0D6H     ;PSW.6 - AUXILIARY CARRY FLAG
CY       BIT     0D7H     ;PSW.7 - CARRY FLAG
```

80C41FA/83C51FA(80C252/83C252)

```
CCF0     BIT     0D8H     ;CCON.0 -PCA MODULE 0 INTERRUPT FLAG
CCF1     BIT     0D9H     ;CCON.1 -PCA MODULE 1 INTERRUPT FLAG
CCF2     BIT     0DAH     ;CCON.2 -PCA MODULE 2 INTERRUPT FLAG
CCF3     BIT     0DBH     ;CCON.3 -PCA MODULE 3 INTERRUPT FLAG
CCF4     BIT     0DCH     ;CCON.4 -PCA MODULE 4 INTERRUPT FLAG
CR       BIT     0DEH     ;CCON.6  COUNTER RUN
CF       BIT     0DFH     ;PCA COUNTER OVERFLOW FLAG
```

8044/8344

```
SER      BIT     0D8H     ;NSNR.0 - RECEIVE SEQUENCE ERROR
NR0      BIT     0D9H     ;NSNR.1 - RECEIVE SEQUENCE COUNTER-BIT 0
NR1      BIT     0DAH     ;NSNR.2 - RECEIVE SEQUENCE COUNTER-BIT 1
NR2      BIT     0DBH     ;NSNR.3 - RECEIVE SEQUENCE COUNTER-BIT 2
SES      BIT     0DCH     ;NSNR.4 - SEND SEQUENCE ERROR
NS0      BIT     0DDH     ;NSNR.5 - SEND SEQUENCE COUNTER-BIT 0
NS1      BIT     0DEH     ;NSNR.6 - SEND SEQUENCE COUNTER-BIT 1
NS2      BIT     0DFH     ;NSNR.7 - SEND SEQUENCE COUNTER-BIT 2
```

80515/80535

```
MX0     BIT     0D8H    ;ADCON.0 - ANALOG INPUT CH SELECT BIT 0
MX1     BIT     0D9H    ;ADCON.1 - ANALOG INPUT CH SELECT BIT 1
MX2     BIT     0DAH    ;ADCON.2 - ANALOG INPUT CH SELECT BIT 2
ADM     BIT     0DBH    ;ADCON.3 - A/D CONVERSION MODE
BSY     BIT     0DCH    ;ADCON.4 - BUSY FLAG
CLK     BIT     0DEH    ;ADCON.5 - SYSTEM CLOCK ENABLE
BD      BIT     0DFH    ;ADCON.7 - BAUD RATE ENABLE
```

80C652/83C652

```
CR0     BIT     0D8H    ;S1CON.0 - CLOCK RATE 0
CR1     BIT     0D9H    ;S1CON.1 - CLOCK RATE 1
AA      BIT     0DAH    ;S1CON.2 - ASSERT ACKNOWLEDGE
SI      BIT     0DBH    ;S1CON.3 - SIO1 INTERRUPT BIT
STO     BIT     0DCH    ;S1CON.4 - STOP FLAG
STA     BIT     0DDH    ;S1CON.5 - START FLAG
ENS1    BIT     0DEH    ;S1CON.6 - ENABLE SIO1
```

80C152/83C152

```
DMA     BIT     0D8H    ;TSTAT.0 - DMA SELECT
TEN     BIT     0D9H    ;TSTAT.1 - TRANSMIT ENABLE
TFNF    BIT     0DAH    ;TSTAT.2 - TRANSMIT FIFO NOT FULL
TDN     BIT     0DBH    ;TSTAT.3 - TRANSMIT DONE
TCDT    BIT     0DCH    ;TSTAT.4 - TRANSMIT COLLISION DETECT
UR      BIT     0DDH    ;TSTAT.5 - UNDERRUN
NOACK   BIT     0DEH    ;TSTAT.6 - NO ACKNOWLEDGE
LNI     BIT     0DFH    ;TSTAT.7 - LINE IDLE
HBAEN   BIT     0E8H    ;RSTAT.0 - HARDWARE BASED ACKNOWLEDGE EN
GREN    BIT     0E9H    ;RSTAT.1 - RECEIVER ENABLE
RFNE    BIT     0EAH    ;RSTAT.2 - RECEIVER FIFO NOT EMPTY
RDN     BIT     0EBH    ;RSTAT.3 - RECEIVER DONE
CRCE    BIT     0ECH    ;RSTAT.4 - CRC ERROR
AE      BIT     0EDH    ;RSTAT.5 - ALIGNMENT ERROR
RCABT   BIT     0EEH    ;RSTAT.6 - RCVR COLLISION/ABORT DETECT
OR      BIT     0EFH    ;RSTAT.7 - OVERRUN
PGSRV   BIT     0F8H    ;IPN1.0 - GSC RECEIVE VALID
PGSRE   BIT     0F9H    ;IPN1.1 - GSC RECEIVE ERROR
PDMA0   BIT     0FAH    ;IPN1.2 - DMA CHANNEL REQUEST 0
PGSTV   BIT     0FBH    ;IPN1.3 - GSC TRANSMIT VALID
PDMA1   BIT     0FCH    ;IPN1.4 - DMA CHANNEL REQUEST 1
PGSTE   BIT     0FDH    ;IPN1.5 - GSC TRANSMIT ERROR
```

80C452/83C452

```
OFRS    BIT     0E8H    ;SLCON.0 - OUTPUT FIFO CH REQ SERVICE
IFRS    BIT     0E9H    ;SLCON.1 - INPUT FIFO CH REQ SERVICE
FRZ     BIT     0EBH    ;SLCON.3 - ENABLE FIFO DMA FREEZE MODE
ICOI    BIT     0ECH    ;SLCON.4 - GEN INT WHEN IMMEDIATE COMMAND
                                   OUT REGISTER IS AVAILABLE
ICII    BIT     0EDH    ;SLCON.5 - GEN INT WHEN A COMMAND IS
                                   WRITTEN TO IMMEDIATE COMMAND IN REG
OFI     BIT     0EEH    ;SLCON.6 - ENABLE OUTPUT FIFO INTERRUPT
IFI     BIT     0EFH    ;SLCON.7 - ENABLE INPUT FIFO INTERRUPT
EFIFO   BIT     0F8H    ;IEP.0 - FIFO SLAVE BUS I/F INT EN
PDMA1   BIT     0F9H    ;IEP.1 - DMA CHANNEL REQUEST 1
PDMA0   BIT     0FAH    ;IEP.2 - DMA CHANNEL REQUEST 0
EDMA1   BIT     0FBH    ;IEP.3 - DMA CHANNEL 1 INTERRUPT ENABLE
EDMA0   BIT     0FCH    ;IEP.4 - DMA CHANNEL 0 INTERRUPT ENABLE
PFIFO   BIT     0FDH    ;IEP.5 - FIFO SLAVE BUS I/F INT PRIORITY
```

80C451/83C451

```
IBF     BIT     0E8H    ;CSR.0 - INPUT BUFFER FULL
OBF     BIT     0E9H    ;CSR.1 - OUTPUT BUFFER FULL
IDSM    BIT     0EAH    ;CSR.2 - INPUT DATA STROBE
OBFC    BIT     0EBH    ;CSR.3 - OUTPUT BUFFER FLAG CLEAR
MA0     BIT     0ECH    ;CSR.4 - AFLAG MODE SELECT
MA1     BIT     0EDH    ;CSR.5 - AFLAG MODE SELECT
MB0     BIT     0EEH    ;CSR.6 - BFLAG MODE SELECT
MB1     BIT     0EFH    ;CSR.7 - BFLAG MODE SELECT
```

83C751/83C752

```
CTO       BIT(READ) 0D8H       ;I2CFG.0 - CLOCK TIMING 0
CT1       BIT(READ) 0D9H       ;I2CFG.1 - CLOCK TIMING 1
T1RUN     BIT(READ) 0DCH       ;I2CFG.4 - START/STOP TIMER 1
MASTRQ    BIT(READ) 0DEH       ;I2CFG.6 - MASTER I2C
SLAVEN    BIT(READ) 0DFH       ;I2CFG.7 - SLAVE I2C
CTO       BIT(WRITE)0D8H       ;I2CFG.0 - CLOCK TIMING 0
CT1       BIT(WRITE)0D9H       ;I2CFG.1 - CLOCK TIMING 1
TIRUN     BIT(WRITE)0DCH       ;I2CFG.4 - START/STOP TIMER 1
CLRTI     BIT(WRITE)0DDH       ;I2CFG.5 - CLEAR TIMER 1 INTERRUPT FLAG
MASTRQ    BIT(WRITE)0DEH       ;I2CFG.6 - MASTER I2C
SLAVEN    BIT(WRITE)0DFH       ;I2CFG.7 - SLAVE I2C
RSTP      BIT(READ) 0F8H       ;I2STA.0 - XMIT STOP CONDITION
RSTR      BIT(READ) 0F9H       ;I2STA.1 - XMIT REPEAT STOP COND.
MAKSTP    BIT(READ) 0FAH       ;I2STA.2 - STOP CONDITION
MAKSTR    BIT(READ) 0FBH       ;I2STA.3 - START CONDITION
XACTV     BIT(READ) 0FCH       ;I2STA.4 - XMIT ACTIVE
XDATA     BIT(READ) 0FDH       ;I2STA.5 - CONTENT OF XMIT BUFFER
RIDLE     BIT(READ) 0FEH       ;I2STA.6 - SLAVE IDLE FLAG
```

80C552/83C552

```
CR0       BIT       0D8H       ;S1CON.0 - CLOCK RATE 0
CR1       BIT       0D9H       ;S1CON.1 - CLOCK RATE 1
AA        BIT       0DAH       ;S1CON.2 - ASSERT ACKNOWLEDGE
SI        BIT       0DBH       ;S1CON.3 - SERIAL I/O INTERRUPT
STO       BIT       0DCH       ;S1CON.4 - STOP FLAG
STA       BIT       0DDH       ;S1CON.5 - START FLAG
ENS1      BIT       0DEH       ;S1CON.6 - ENABLE SERIAL I/O
ECT0      BIT       0E8H       ;IEN1.0 - ENABLE T2 CAPTURE 0
ECT1      BIT       0E9H       ;IEN1.1 - ENABLE T2 CAPTURE 1
ECT2      BIT       0EAH       ;IEN1.2 - ENABLE T2 CAPTURE 2
ECT3      BIT       0EBH       ;IEN1.3 - ENABLE T2 CAPTURE 3
ECM0      BIT       0ECH       ;IEN1.4 - ENABLE T2 COMPARATOR 0
ECM1      BIT       0EDH       ;IEN1.5 - ENABLE T2 COMPARATOR 1
ECM2      BIT       0EEH       ;IEN1.6 - ENABLE T2 COMPARATOR 2
ET2       BIT       0EFH       ;IEN1.7 - ENABLE T2 OVERFLOW
PCT0      BIT       0F8H       ;IP1.0 - T2 CAPTURE REGISTER 0
PCT1      BIT       0F9H       ;IP1.1 - T2 CAPTURE REGISTER 1
PCT2      BIT       0FAH       ;IP1.2 - T2 CAPTURE REGISTER 2
PCT3      BIT       0FBH       ;IP1.3 - T2 CAPTURE REGISTER 3
PCM0      BIT       0FCH       ;IP1.4 - T2 COMPARATOR 0
PCM1      BIT       0FDH       ;IP1.5 - T2 COMPARATOR 1
PCM2      BIT       0FEH       ;IP1.6 - T2 COMPARATOR 2
PT2       BIT       0FFH       ;IP1.7 - T2 OVERFLOW
```

80C517/80C537

```
F1        BIT       0D1H       ;PSW.1 - FLAG 1
MX0       BIT       0D8H       ;ADCON0.0 - ANALOG INPUT CH SELECT BIT 0
MX1       BIT       0D9H       ;ADCON0.1 - ANALOG INPUT CH SELECT BIT 1
MX2       BIT       0DAH       ;ADCON0.2 - ANALOG INPUT CH SELECT BIT 2
ADM       BIT       0DBH       ;ADCON0.3 - A/D CONVERSION MODE
BSY       BIT       0DCH       ;ADCON0.4 - BUSY FLAG
CLK       BIT       0DEH       ;ADCON0.5 - SYSTEM CLOCK ENABLE
BD        BIT       0DFH       ;ADCON0.7 - BAUD RATE ENABLE
```

80C154/83C154

```
ALF       BIT       0F8H       ;IOCON.0 - CPU POWER DOWN MODE CONTROL
P1F       BIT       0F9H       ;IOCON.1 - PORT 1 HIGH IMPEDANCE
P2F       BIT       0FAH       ;IOCON.2 - PORT 2 HIGH IMPEDANCE
P3F       BIT       0FBH       ;IOCON.3 - PORT 3 HIGH IMPEDANCE
IZC       BIT       0FCH       ;IOCON.4 - 10K TO 100 K OHM SWITCH (P1-3)
SERR      BIT       0FDH       ;IOCON.5 - SERIAL PORT RCV ERROR FLAG
T32       BIT       0FEH       ;IOCON.6 - 32 BIT TIMER SWITCH
WDT       BIT       0FFH       ;IOCON.7 - WATCHDOG TIMER CONTROL
```

Appendix C: Supported Parts And Emulation

Probe Card Support Guide

Applications may be developed for a microcontroller other than the specific controller supported by your probe card. For example, an 8052 probe card can be used to develop 8031 applications as long as you do not attempt to use Port 0 or Port 2 (the address/data bus for the 8031) for I/O. An 8032 probe card can be used to develop 8031 applications as long as the program does not make use of Timer 2 or the additional 128 bytes of internal data RAM.

The SFR's, the key to emulation compatibility, were originally defined in the 8051. In later proliferation psrts, bytes that were unused in the 8051 were defined to support and control new features or functions such as new ports and analog to digital converters. The original SFR addresses are rarely redefined.

The following list shows which microcontrollers are supported by each of the available probe cards:

8052 Probe Card

8031	8032	8051	8052	8053	80C154	80C32T2
80C31	80C32	80C51	80C52	8753	83C154	80C52T2
		8751	8752		85C154	
		87C51	87C52			

80C51FA Probe Card

8031	80C31	8032	80C32	80C51FA	80C32T2

80C154 Probe Card

8031	80C31	8032	80C32	80C154	80C32T2

80C521 Probe Card

8031	8051	8053	80C321	80C541	80C32T2
80C31	80C51	8753	80C521	87C541	80C32T2
		8751		87C521	
		87C51			

80C321 Probe Card

8031	80C31	80C321	80C32T2

8031 Probe Card

8031	80C31

8032 Probe Card

8031	8032	80C31	80C32	80C32T2

8344 Probe Card

8344

83C751 Probe Card

83C751	87C751

83C752 Probe Card

83C752	87C752

80C535 Probe Card

80535	80C535

80532 Probe Card

80532

80C537 Probe Card

80C537

80C451 Probe Card

80C451

83C451 Probe Card

80C451 83C451 87C451

80C552 Probe Card

80C552

83C552 Probe Card

80C552 83C552 87C552

80C652 Probe Card

8031 80C31 80C32T2 80C652

83C152 Probe Card

80C152JA/JC 83C152JA/JC

80C152JB/JD (limited)

80C851 Probe Card

8031 80C31 80C851

83C053 Probe Card

83C053 83C054 87C054

80CL410 Probe Card

80CL410

80515 Probe Card

80515 80C515 80535 80C535

80C517 Probe Card

80C517 80C537

83C550 Probe Card

80C550 83C550 87C550

83C652 Probe Card

80C652 83C652 87C652

83C654 Probe Card

80C652 83C654 87C654

Appendix D: Character Sets

Single Line Assembler Character Set

The legal character set for the Single Line Assembler (*Display/Alter|Asm/Dasm|Assemble*, page 7-37) is made up of the following characters (listed in ascending ASCII order):

$ ' ( ) * + , - . / 0 1-9 : ? @ A-Z _ a-z

Fill Search Pattern Character Set

The legal character set for the fill pattern specification (*Display/Alter|Code|Fill*, *Display/Alter|Idata|Fill* and *Display/Alter|Xdata|Fill*, (page 7-38) is made up of the following characters (listed in ascending ASCII order):

' 0-9 ? A-Z _ a-z

Appendix E: Using Symbols

Symbolic Debug Support Features

The emulator host software includes specific support for programs written in high level languages (e.g., Intel PL/M-51, Archimedes C, Franklin C, etc.). Features include:

- Multi-Module Support
  - Handling of local symbols
- Source Line Numbers
  - Source line number included in disassembly (Code Disassembly and Main Source Window)
  - Simple Breakpoints by source line number (single line number or a range of line numbers)
  - Single-Step by source line number (within a module or across entire program)
  - Single-Step by procedure/function entry points
  - Go (resume execution) at a particular source line number
- Ranges in Simple Breakpoints ("<from_addr> - <to_addr>")

Note that when debugging C code, several compilers (such as Archimedes C) are stack oriented in that local automatic variables are referenced using the stack and cannot be referenced symbolically. If you need to reference this type of variable symbolically during debugging, they should be declared as 'static'.

Syntax For Symbolic Input

Symbols (Code Labels And Variable Names)

In all cases where you may specify a name (code label or variable name) from the keyboard, the following syntactic forms are recognized/required. Each form is followed by a description of what the host software does to locate the appropriate definition of the symbol:

<name>

First search for <name> in local symbols of the current module; If not found, then search for <name> in the program's global symbols.

USAGE: to designate a global symbol or a local symbol defined immediately within the current module.

<Module_Name>:<name>

Search for <name> in local symbols of module <Module_Name>.

USAGE: to designate a local symbol defined immediately within the module <Module_Name>.

<Module_Name>:<Procedure_Name>:<name>

Search for <name> in the local symbols of procedure/function <Procedure_Name> in module <Module_Name>.

USAGE: to designate a local symbol defined immediately within the procedure/function <Procedure_Name> which is located in module <Module_Name>.

Note that this form may be shortened to <Procedure_Name>:<name> if the <Procedure_Name>, even though it may be a local symbol within the module, is not repeated (duplicated) in another module. For example, if we want to reference the local variable J defined in procedure PROC_X which is contained in module MOD_M, we could specify **MOD_M:PROC_X:J**. If PROC_X is not defined as a module name or as a local procedure/function in any other module in the program, we could shorten the specification to **PROC_X:J**. Furthermore, if it happens that this local symbol J is not defined in any other procedure or module of the program, then we could simply specify **J**.

Line Numbers

At any place where a code address or label can be specified, a line number can also be specified (e.g., disassembly start address, simple breakpoint, etc.). Line numbers are specified as follows:

#<n>

designates source line number <n> in the current module.

<Module_Name>:#<n>

designates source line number <n> in module <Module_Name>.

Examples Of Absolute And Symbolic Input

0A7h	absolute address (hexadecimal)
132	absolute address (decimal)
START	label or variable START defined as a global (public) symbol or a local symbol in the current module
MODA:RESTART	label or variable RESTART which is defined as a local symbol in module MODA
#45	source line number 45 in the current module.
MODA:#14	source line number 14 in module MODA
MODA:PROC:ARG	label or variable ARG which is defined as a local symbol within the procedure/function PROC which is contained within the module MODA
MODA:PROC:FUNC:I	denotes the local variable I defined within function FUNC which is contained within procedure PROC which is contained within module MODA (applies only to PL/M-51 as the C language does not allow nested function definitions)

Appendix F: File Formats

The emulator Host software supports three basic object file formats; the Intel Hex format, the Motorola S-Record format and the Intel Absolute Object Module format (commonly called AOMF). You do not need to distinguish between them when loading a program. Each is described below:

Intel Hex / Motorola S-Record Object File (PROMable)

This is an ASCII file format which is normally used as input to PROM programming utilities. In its "pure form", a Hex file contains no symbolic debug information.

However, extensions to this file format are supported by various third parties (e.g., Microtec Research) in which a limited amount of symbolic information is present. The general format of a Hex or S-Record file is as follows:

```
Symbolic Information Format (optional, at head of file) (Microtec, BSO))

NOTE: This format does not conform specifically/exactly to anyone's
(e.g., Microtec's, BSO's, etc.) current format for symbolic
information. The specification given here (which is what the
Host Software actually supports during program loading) is a
*superset* of all those variations.
     ____
    |  __ <ws*><$_or_$$><ws*><mod_name_opt><ws*><newline> {1st optional}
    | |   <ws*><num_opt><ws*><memsp_opt><ws*><symbol><ws+><addr>
   1| |   |_________________________________________________________|
    |2|      |_ Sequence may repeat any number of times within a single re-
cord.
    | |
    | |   <ws*><num_opt><ws*><memsp_opt><ws*><symbol><ws+><addr>
    | |      ...
    | |__ <ws*><num_opt><ws*><memsp_opt><ws*><symbol><ws+><addr>
    |____
          ...
          <ws*><$_or_$$><ws*><newline>   (symbolic info terminator)

where <ws*>       == Zero or more whitespace
                     (blank or horizontal tab) characters.
      <ws+>       == One or more whitespace
                     (blank or horizontal tab) characters.
      <$_or_$$>   == A single dollar sign character ('$')
                     or two dollar sign characters ('$$').
      <num_opt>   == A single zero character ('0'),
                     or a decimal/octal number,
                     or nothing.  (ignored)
      <memsp_opt>== Optional memory space designator number:
                       0 Code ROM/PROM (simple label, proc/func name)
                         {applicable to MCS-51,68HC11,COP8}
                       1 External data RAM
                         {applicable to MCS-51}
```

```
                       2 Direct   (internal directly-addressable RAM)
                         {applicable to MCS-51,COP8}
                       3 Indirect (internal indirectly-addressable RAM)
                         {applicable to MCS-51}
                       4 Bit      (internal bit-addressable RAM)
                         {applicable to MCS-51}
                       5 Number   (e.g., assembly language EQU)
                         {applicable to MCS-51,68HC11,COP8}
                     If this designator is present, <num_opt> must
                     also be supplied (and the <ws*> between <num_opt>
                     and <memsp_opt> is actually <ws+>).
                     68HC11 NOTE: If <memsp_opt> is 2 (internal
                     directly-addressable RAM), it will be changed to
                     0 (code memory) before insertion into the internal
                     Symbol Table. This is done because the 68HC11 has
                     a single, uniformly-addressable 64K memory space
                     for both code and data.
         <addr>  ==  Symbol's address/value: Zero to 5 characters from
                     the sets '0'-'9', 'A'-'F', 'a'-'f'. The symbol's
                     address/value will be treated as a hexadecimal
                     number. The <addr> may be terminated by an
                     optional radix character, 'H' or 'h'.
      <newline>  ==  End-of-record.
```

1. If a module name is supplied, subsequent symbol definitions will be treated as local symbols within that module. Otherwise, subsequent symbol definitions will be treated as global (public) symbol definitions.
2. One or more symbol records (each containing one or more symbol definitions) per module.

The '$' or '$$' record separating the symbol records from the code records is optional for Intel Hex files (':'), but required for Motorola S-Record files.

<u>Code Format: Intel Hex Format</u>

0	1	2	3	4	5	6	7	8	9	10	11	12	---	xx	xx
:	bytes		address				type		data1		data2		---	cksum	

NOTE: No whitespace is allowed preceding the ':' at the beginning of each text record.

1. 'cksum' does not including the leading ':' character. All remaining bytes, including the checksum byte, are added; the result must be zero.
2. 'bytes' field is a count of the 'datan' pairs only.

<u>Code Format: Motorola S-Record Format</u>

0 1   2 3   4 5 6 7   xx xx

Type	Record Length	Address	Code/Data	Checksum
		(aaaa)		
		(aaaaaa)		
(Sn)	(ll)	(aaaaaaaa)	(dd...dd)	(kk)

```
ll ==   length field (00-FF) and is a count of character pairs
        which follow in this S-Record (including the checksum).
kk ==   checksum (least significant byte of the one's complement of
        the sum of the values represented by the pairs of characters
        making up the record length, address and code/data fields).

    S0 ll aaaa <opt> kk
        Header record for a block of S-Records.
        aaaa (address field) is normally zeroes.
    S1 ll aaaa dd...dd kk
        Text (Code/Data Init), 2-byte load address.
*   S2 ll aaaaaa dd...dd kk
        Text (Code/Data Init), 3-byte load address.
*   S3 ll aaaaaaaa dd...dd kk
        Text (Code/Data Init), 4-byte load address.
*   S4 ll ??? kk
        ?
*   S5 ll nnnn kk
        Count of S1,S2 and S3 records (transmitted)
        in a particular block.
        'nnnn' == count (appears in address field)
*   S6 ll ??? kk
        ?
*   S7 ll aaaaaaaa kk
        Termination Record for a block of S3 records.
        aaaaaaaa == optional 4-byte entry address
*   S8 ll aaaaaa kk
        Termination Record for a block of S2 records.
        aaaaaa   == optional 3-byte entry address
    S9 ll aaaa kk
        Termination Record for a block of S1 records.
        aaaa     == optional 2-byte entry address

NOTE: No whitespace is allowed preceding the"Sn" at the beginning of
each text record.

* == These record types are not supported during loading.

GENERAL NOTE: We also allow intermixing of Intel Hex Format records and
Motorola S-Record Format Records.
```

Intel Absolute Object Module Format (AOMF) File

This is a binary file format which is composed of, at a minimum, object records containing code to be loaded. Depending on which compilation and link options are used (see Appendix H, Recommended Compilation Options), there may also be object records containing symbolic information such as labels, variable names, module scope definitions, line numbers and variable type information.

Appendix G: HLL Support Of Third Party Software

Source Level Debug Support

The Host Software includes support for debugging HLL (High-Level Language) programs (e.g., C and PL/M-51) at the source level. If the program loaded into code memory contains the appropriate debug records (see Appendix H, Recommended Compilation Options), the Host Software is able to display HLL source in traces, disassemblies and other contexts.

Any time a program is loaded using the *File | Load* command (page 7-27) or *File | Restore* command (page 7-28), the Host Software attempts to locate the source file corresponding to each module in that program in the directory specified by the HLL search path. The path can be set in one of three ways:

1) (statically) on command line invocation of software via the '-s <source_path>' option (see Appendix J)
2) (dynamically) using the *Source/Symbols | Source Path* command (page 7-52)
3) (automatically) if the path has not been set via the command line option, the path is set to the current directory at initialization

Once the path is set it will not change unless changed dynamically using the *Source/Symbols | Source Path* command. In fact, changing the current working directory during an escape to DOS (*File | OS-Escape*) has no effect on the path.

If the program contains source line number debug records (see Recommended Compilation Options), the Host Software will "remember" the position of each source image for which there is a line number debug record entry. This is done so that HLL displays can be generated quickly.

During the search for source files, messages are displayed on the Quick Help Line to inform the user of the Host Software's progress. The length of time these messages are displayed (from zero to thirty seconds) is controlled by model file directive A36 (Appendix M).

For each module containing line number debug records for which the corresponding source file is found, the following message is displayed:

Source Found For: <module_name>

If the source file was found in the current directory, as opposed to being found in the directory specified in the HLL search path, the <module_name> in the above message will be preceded by an asterisk.

If the program contains module and line number debug records, but the Host Software is unable to locate any source file (for at least one module in the program), the following message is displayed:

Source Line Numbers Present, but No Source Files Found

If this message appears when source files are available (and source-level debug is desired), make sure that the HLL search path is set correctly.

If the program contains no module or line number debug records, the following message is displayed:

No Source Line Numbers Present

If this message appears when source files are available (and source-level debug is desired), make sure that the modules in the program were compiled with the appropriate debug options, and that the program was linked with the appropriate debug options.

Note that this message always will appear, if enabled, for programs composed only of assembly language modules.

Locating Source Files

The Host Software must be able to locate the source file(s) for the module(s) in a program given only the module name(s). The module names are all that is provided in the debug records in the absolute object module format file. Given a module name, <module_name>, the Host Software will search (through the directories/paths described above), for the first file named:

<module_name>.C

If such a file is found, that file is assumed to contain "raw" C source code. C compilers normally supply a module name in the object module's debug records which is the same as the source file name without the '.C' suffix. It is from this '.C' file that Host Software obtains the source images displayed in disassemblies and traces.

<module_name>.LST

If such a file is found, that file is assumed to contain the source listing generated by the PL/M-51 compiler. The emulator Host Software must read this file, rather than the original PL/M-51 source file, because the "line numbers" provided by the PL/M-51 compiler in the object module's debug records are not really line numbers at all, but rather statement numbers, as determined by the PL/M-51 compiler. These statement numbers are also present in the '.LST' file generated by the PL/M-51 compiler. It is from this '.LST' file that the Host Software obtains the source images displayed in disassemblies and traces.

Note that the debug records in an absolute object module format file give no explicit indication of a module's original source language, nor do they specify the program (assembler or compiler) which created the relocatable object module corresponding to a module in the linked program.

Display Formatting Of Source Images

The user has control over the following aspects of the display format of HLL source images in all menus:

1) Whether or not to skip (ignore) leading blanks in each source image. The default action is not to skip leading blanks.

2) The tab stop settings (column width) to be used for the ASCII horizontal tab character (09 hex). The default tab stop setting is 4 (i.e., a 4-character column width between tab stops).

For more details see directive A41 in Appendix M, Model File Configuration.

These characteristics are determined by the particular language processor and its implementation. They affect the Host Software's capabilities, but are beyond the Host Software's control.

Archimedes C-51 Release V.2.01

1) Line numbers correspond to source file line numbers (relative source record numbers).
2) The run-time library contains debug symbol tables for each entry point for each module in the library. Thus, in single-stepping by procedure/function entry points, the run-time library routines are not transparent; a break will occur on entry to each run-time library routine.
3) The run-time library does not contain debug line number tables for the modules in the library. Thus, in single-stepping by line number across all modules in the program, the run-time library routines are transparent; no breaks will occur in run-time library routines.

Franklin C51 (V2.12) With L51 (V2.4)

1) Line numbers correspond to source file line numbers (relative source record numbers).
2) The run-time library contains debug symbol tables for each entry point for each module in the library. Thus, in single-stepping by procedure/function entry points, the run-time library routines are not transparent; a break will occur on entry to each run-time library routine.
3) The run-time library does not contain debug line number tables for the modules in the library. Thus, in single-stepping by line number across all modules in the program, the run-time library routines are transparent; no breaks will occur in run-time library routines.
4) There is a local code label (_ICE5100_BUG_) generated by the compiler for each module created by the user. This symbol will cause no problems.
5) Line numbers corresponding to the end of a FOR loop are not as one would expect. It seems that the line number of the FOR statement itself is also used for the last line of the FOR loop. This is due to the nature of the line number tables generated by the C51 compiler and cannot be remedied by the Host Software.

   The code in Figure 1 shows a line number table created with this quirk.

```
 17  1            for (i = 0; i < repeat; i++) {
 18  2               wastetime();
 19  2               if (P1_0)
 20  2                  P1_0 = 0;
 21  2               else
 22  2                  P1_0 = 1;
 23  2               state = 0;
 24  2               if (P1_0)
 25  2                  state= (state)? 0 : 1;
 26  2               } /* end of: for 'i' */
 27  1            } /* end of function */

                                            ; SOURCE LINE # 17
0000  E4              CLR     A
0001  F500     R      MOV     i,A
0003           ?C0001:
0003  C3              CLR     C
0004  E500     R      MOV     A,i
0006  9500     R      SUBB    A,repeat_cnt
0008  A2E7            MOV     C,ACC.7
000A  30D201          JNB     OV,?C0009
000D  B3              CPL     C
000E           ?C0009:
000E  5022            JNC     ?C0002
                                            ; SOURCE LINE # 18
0010  120000   E      LCALL   wastetime
                                            ; SOURCE LINE # 19
0013  309704          JNB     P1_0,?C0004
                                            ; SOURCE LINE # 20
0016  C297            CLR     P1_0
0018  8002            SJMP    ?C0005
001A           ?C0004:
                                            ; SOURCE LINE # 22
001A  D297            SETB    P1_0
001C           ?C0005:
                                            ; SOURCE LINE # 23
001C  E4              CLR     A
001D  F500     E      MOV     state,A
                                            ; SOURCE LINE # 24
001F  30970C          JNB     P1_0,?C0003
                                            ; SOURCE LINE # 25
0022  E500     E      MOV     A,state
0024  6004            JZ      ?C0007
0026  7F00            MOV     R7,#00H
0028  8002            SJMP    ?C0008
002A           ?C0007:
002A  7F01            MOV     R7,#01H
002C           ?C0008:
002C  8F00     E      MOV     state,R7
002E           ?C0003:
                                            ; SOURCE LINE # 17
002E  0500     R      INC     i
0030  80D1            SJMP    ?C0001
0032           ?C0002:
0032  22              RET
```

Figure G-1. Franklin C51 - FOR Loop

1) Line numbers are actually statement numbers, as determined by the compiler. These statement numbers appear as the first column of numbers in the source listing generated by the compiler (LIST option). The statement numbers also appear as comments in the generated object code listing (CODE option).

2) The run-time library (PLM51.LIB) contains neither debug symbol tables nor debug line number tables for the modules in the library. Thus, in single-stepping by line number across all modules in the program or in single-stepping by procedure/function entry points, the run-time library routines are transparent; no breaks will occur in run-time library routines.

3) The apparent behavior of single-stepping by line number across an IF-THEN-ELSE statement is not exactly what one would expect. If the ELSE branch of an IF-THEN-ELSE statement is taken, control will appear to transfer from evaluation of the IF condition to the statement following the ELSE statement.

   If the ELSE statement is a compound statement (simple DO block), control will appear to transfer ("correctly") from evaluation of the IF condition to the first statement in the ELSE block.

   This is due to the nature of the line number tables generated by the PL/M-51 compiler and cannot be remedied by the Host Software.

   Examine the code in Figure 2 generated for a simple IF-THEN-ELSE statement and in Figure 3 generated for a compound IF-THEN-ELSE statement and the location of the line number comments; these comments accurately reflect the actual content of the line number table generated by the compiler for the module:

```
28  2              IF sw$b THEN
29  2                 CALL proc_b1;
30  2              ELSE
                      CALL proc_b2;
31  2              RETURN arg;
```

```
                                          ; STATEMENT # 28
002B  300005 F            JNB   SWB,THEN?5
                                          ; STATEMENT # 29
002E  120000 F            LCALL PROC_B1
                                          ; STATEMENT # 30
0031  8003                SJMP  ELSE?6
0033            THEN?5:
0033  120000 F            LCALL PROC_B2
                                          ; STATEMENT # 31
0036            ELSE?6:
0036  E500   F            MOV   A,ARG+0001H
0038  22                  RET
```

Figure G-2. Intel PL/M-51 - Simple IF-THEN-ELSE

```
28  2                IF sw$b THEN
29  2                    CALL proc_b1;
30  2                ELSE
                         DO;
31  3                    CALL proc_b2;
32  3                    CALL proc_b2;
33  3                    CALL proc_b2;
34  3                    END;
35  2                RETURN arg;

                                      ; STATEMENT # 28
002B  300005 F         JNB   SWB,THEN?5
                                      ; STATEMENT # 29
002E  120000 F         LCALL PROC_B1
                                      ; STATEMENT # 30
0031  8009             SJMP  ELSE?6
0033          THEN?5:
                                      ; STATEMENT # 31
0033  120000 F         LCALL PROC_B2
                                      ; STATEMENT # 32
0036  120000 F         LCALL PROC_B2
                                      ; STATEMENT # 33
0039  120000 F         LCALL PROC_B2
                                      ; STATEMENT # 35
003C          ELSE?6:
003C  E500   F         MOV   A,ARG+0001H
003E  22               RET
```

Figure G-3. Intel PL/M-51 - Compound IF-THEN-ELSE

Micro Computer Control Corporation (MCC) MICRO/C-51 Release V1.3A

1) The run-time library contains neither debug symbol tables nor debug line number tables for the modules in the library. Thus, in single-stepping by line number across all modules in the program or in single-stepping by procedure/function entry points, the run-time library routines are transparent; no breaks will occur in run-time library routines.

1) Line numbers are actually statement numbers, as determined by the compiler. These statement numbers appear as the first column of numbers in the source listing generated by the compiler ('-p' option). The statement numbers also appear as comments in the generated assembly source code file.

2) The run-time library contains debug symbol tables for each entry point for each module in the library. Thus, in single-stepping by procedure/function entry points, the run-time library routines are not transparent; a break will occur on entry to each library routine.

3) The run-time library does not contain debug line number tables for the modules in the library. Thus, in single-stepping by line number across all modules in the program, the run-time library routines are transparent; no breaks will occur in run-time library routines.

4) If a PL/M-51 source statement is spread over more than one line a line number table entry is output for only the last line of the range. This is due to the nature of the line number tables generated by the PL/M-51 compiler and cannot be remedied by the Host Software. The code in Figure 4 shows an example of this behavior.

```
  17  2                    sw$main = true;
  18  2                    DO k = 9 TO 11 BY 1;
  19  3                        j = (arg + k)
  20  3                            *
  21  3                            5;
  22  3                        END; /* k */;
```

```
DM_MODA:#17:        sw$main = true;
01B8  D201    _FUNC_A:    SETB   _SWMAIN   ;DM_MODA:#17
DM_MODA:#18:        DO k = 9 TP 11 BY 1;
01BA  752C09              MOV    _1,#09h   ;DM_MODA:#18
01BD  E52C    _3:         MOV    A,_1
01BF  D3                  SETB   C
01C0  940B                SUBB   A,#0Bh
01C2  501F                JNC    _4
DM_MODA:#21                 5;
01C4  7F05    _5:         MOV    R7,#05h   ;DM_MODA:#21
01C6  7E00                MOV    R6,#00h
01C8  AB2C                MOV    R3,_1
01CA  7A00                MOV    R2,#00h
01CC  EB                  MOV    A,R3
01CD  2530                ADD    A,30h
```

Figure G-4. Tasking PL/M-51 - Multiple Line Statement

Appendix H: Recommended Compilation Options

Archimedes Products

1) Archimedes 'a8051' (relocatable assembler for MCS-51):

S	Put all symbols (local and global) into object module

2) Archimedes 'c-51' (C cross-compiler for MCS-51):

-j	Put static and local symbols into object debug tables for Release V2.01 and earlier.
	or
-rn	Put static and local symbols into object debug tables for Release V3.00B and beyond.

3) Archimedes 'xlink' (linker for MCS-51):

-c8051	Define CPU as 8051
-Faomf8051	Generate Intel Absolute Object Module (Load Module) Format
-xsmi	Generate cross-reference listing (optional)
-z	Disable overlay check (optional)

Franklin Products

1) Franklin 'a51' (relocatable assembler for MCS-51):

$DEBUG	Include debug information in object file

2) Franklin 'c51' (C cross-compiler for MCS-51):

CODE	Append assembly language mnemonics list to '.LST' file (optional)
DEBUG	Include debug (symbols
INTVECTOR	Put instruction at address 0 to 'LJMP' past interrupt vectors
OBJECTEXTEND	Include variable data type definitions in object module

3) Franklin 'l51' (linker for MCS-51):

IXREF	Generate cross reference report (optional)
DEBUGLINES	Include line number information in output file
DEBUGPUBLICS	Include public (global) symbol information in output file
DEBUGSYMBOLS	Include local symbol information in output file

1) Intel 'asm51' (relocatable assembler for MCS-51):

$OBJECT	Output relocatable object module to 'source_file_base_name.OBJ'
$DEBUG	Include debug symbol information in the object module

2) Intel 'plm51' (PL/M-51 compiler for MCS-51):

$DEBUG	Generate debug records in the object module
$LIST	Allow listing of source lines
$PRINT	Output source listing to 'source_file_base_name.LST'
$CODE	Append generated code listing to 'source_file_base_name.LST' (optional)

NOTE: All module names (in all source files) should be the same as the base name of the '.LST' listing file:

```
dm_main:DO; /* beginning of main module (source file name:DM_MAIN.P51) */
....
END dm_main;

dm_moda: DO; /* beginning of module A (source file name: DM_MODA.P51) */
....
END dm_moda;
```

3) Intel 'rl51' (linker for MCS-51):

DEBUGLINES	Include line number information in output file
DEBUGPUBLICS	Include public (global) symbols in output file
DEBUGSYMBOLS	Include local symbols in output file
PRINT	(optional) create link summary (report) file
MAP	(optional) output memory map to link summary
SYMBOLS	(optional) output local symbols to link summary
PUBLICS	(optional) output public (global) symbols to link summary
LINES	(optional) output line (statement) numbers to link summary
IXREF	(optional) output intermodule cross-reference to link summary

MetaLink Products

1) MetaLink 'asm51' (absolute assembler for MCS-51):

$DEBUG	Generate Intel absolute object module format
$OBJECT	Generate Intel HEX format

Micro Computer Control Corporation(MCC) Products

1) MCC MICRO/C-51 'mcc51' (C cross-compiler for MCS-51):

/t	Generate source line number information
/c	Include C source as commentary in generated '.SRC' (assembly language) source file (optional)

2) If using Intel's assembler & linker to assemble generated '.SRC' files and link the resultant relocatable object modules, see the recommended options for **Intel Products** in this appendix.

3) Invoke MCC 'mcsu' (Statement Label Conversion Utility) with the linker-generated Absolute Object Module Format (AOMF) file as input. Note that you should be using MICRO/C- 51 V1.3 or later with the 'mcsu' conversion utility.

1) Tasking 'plmti51' (PL/M-51 compiler for MCS-51):

The output from the compiler is assembly language source code that needs to be assembled by the Tasking assembler.

-p	Output source listing file to 'source_file_base_name.LST'. This is the file that will be used by the host software for High Level Language debugging

NOTE: All module names (in all source files) should be the same as the base name of the '.LST' listing file:

dm_main:DO; /* beginning of main module (source file name:DM_MAIN.PLM) */
....
END dm_main;

dm_moda: DO; /* beginning of module A (source file name:DM_MODA.PLM) */
....
END dm_moda;

2) BSO/Tasking 'cc51' (C compiler for MCS-51):

The output from the compiler is assembly language source code that needs to be assembled by the BSO/Tasking assembler.

-gt	generate the maximum amount of HLL symbolic debug information

3) Tasking 'asm51' (relocatable assembler for MCS-51):

PRINT(filename)	Generate listing file. Be sure to use a filename extension other than '.LST' as the host software will use the '.LST' file generated by the compiler for High Level Language debugging
DB	Generate (enable) debug information
DI(0BBh)	Type of debug information generated (symbols and line numbers)

4) Tasking 'link51' (linker for MCS-51):

DL	Include line number information in output file
DP	Include public symbol information in output file
DS	Include local symbol information in output file

5) The output from the Tasking linker must be converted to the Intel Absolute Object Module Format (AOMF) using the Tasking conversion utility 'oct_o51' in order to be loaded by the Host Software.

Appendix I: Using A Mouse

While iceMASTER currently does not provide internal, direct support for a mouse, you can operate iceMASTER with a mouse by using the "mouse menu" utility normally supplied by each mouse manufacturer. These mouse menu utilities allow you to define a mapping from mouse buttons/movements to keyboard keystrokes.

Regardless of which mouse you are using, mouse movement should always be mapped as:

Mouse Movement	Keystroke Mapping
Left	←
Right	→
Up	↑
Down	↓

You can control iceMASTER most effectively with a mouse if your mouse menu software allows you to define mouse buttons for at least the following keystrokes (2-button mouse):

Mouse Button	Keystroke Mapping
Left	Enter
Right	Esc
Left + Right	Tab

If you have a 3-button mouse, we recommend the following definitions:

Mouse Button	Keystroke Mapping
Left	**Enter**
Right	**Esc**
Middle	**Tab**
Left + Middle	**Home**
Left + Right	**Shift-Tab**
Left + Middle + Right	**Ctrl-E**

Specific details for setting up mouse menus are provided below for:

- Microsoft Mouse
- LOGITECH Mouse
- Mouse Systems White Mouse

If your mouse is not listed above, refer to the mouse manufacturer's documentation.

Additionally, if you decide to use a mouse in this manner, you will probably prefer to set the *Configure | Options | Bad key/command* toggle to OFF (page 7-21).

The following paragraphs show how to set up a mouse menu for iceMASTER using a Microsoft Mouse (2-button).

Assuming that the mouse driver files are located at C:\MSM, the following steps will create a mouse menu which can be loaded to control iceMASTER. The two files defining the mouse button/movement mapping will be created as C:\MSM\MENUS\ICE.DEF and C:\MSM\MENUS\ICE.MNU. To create ICE.MNU:

```
MKDIR C:\MSM\MENUS
CD    C:\MSM\MENUS
C:\MSM\MENUMAKE
                  Edit
                        Buttons:
                              L(eft)                  <Enter>
                              R(ight)                 <Esc>
                              B(oth)                  <Tab>
                        Movement:
                              L(eft)                  <Left>
                              R(ight)                 <Right>
                              U(p)  <Up>
                              D(own)                  <Down>
                  Save As
                        C:\MSM\MENUS\ICE
                  Make Menu
                        ICE.MNU
```

The following is a fragment of an AUTOEXEC.BAT file which automatically loads the mouse driver and the iceMASTER mouse menu driver created above:

```
REM AUTOEXEC.BAT Fragment for Microsoft Mouse:
REM
REM (Tested using Version 7.04 of the Microsoft Mouse Driver)
REM (Microsoft Mouse software files at "C:\MSM\")
REM
REM 1.  Load Mouse Driver:
REM     (COM2, Horizontal & Vertical Sensitivity of 10
REM            (range: 5-100, in increments of 5)      )
C:\MSM\MOUSE /c2 /s10
REM
REM 2.  Use "Unaccelerated" Mouse-Movement Profile:
C:\MSM\SETSPEED /p4 /fC:\MSM\MOUSEPRO.FIL
REM
REM 3.  Load "Control Panel"
REM     (invoke later with Ctrl-Alt-left_mouse_button):
REM C:\MSM\cpanel
REM
REM 4.  Install iceMASTER Menu.
REM     iceMASTER menu files are  C:\MSM\MENUS\ICE.DEF
REM                           and C:\MSM\MENUS\ICE.MNU
REM     This menu defines the mouse buttons as follows:
REM        Buttons:                 Movement:
REM          Left  = Enter            L = Left
REM          Right = Esc              R = Right
REM          Both  = Tab              U = Up
REM                                   D = Down
```

```
C:\MSM\MENU C:\MSM\MENUS\ICE
REM
REM Later, to dynamically remove one or more of the mouse utilities:
REM     CPANEL OFF                          {remove Control Panel}
REM     MENU OFF                            {remove Mouse Menus   }
REM     MOUSE OFF                           {remove Mouse Driver  }
```

LOGITECH Mouse Support

The following paragraphs show how to set up a mouse menu for iceMASTER using a LOGITECH Mouse (3-button).

Assuming that the mouse driver files are located at C:\LTM, the following steps will create a mouse menu which can be loaded to control iceMASTER.

You first need to create a menu source file, ICE.DEF, containing the definitions mapping mouse movements/buttons to keyboard keystrokes. You can use any text editor or word processor to create a plain ("non-document") ASCII text file containing:

```
; LOGIMENU File definition for iceMASTER
; Horizontal Sensitivity:  150/200
; Vertical   Sensitivity:  140/200
BEGIN  leftb, midb, rightb, leftm, rightm, upm, downm, 150, 140
CHORDS lmb, lrb, mrb, allb
;
; Mouse Buttons:
;
leftb:  TYPE ENTER ;Left   Button = Enter key
midb:   TYPE TAB   ;Middle Button = Tab   key
rightb: TYPE ESC   ;Right  Button = Esc   key
;
; Mouse Movement:
;
leftm:  TYPE 0,75  ;Left   Movement = Left  arrow key
rightm: TYPE 0,77  ;Right  Movement = Right arrow key
upm:    TYPE 0,72  ;Up     Movement = Up    arrow key
downm:  TYPE 0,80  ;Down   Movement = Down  arrow key
;
; Chords (multi-button press):
;
lmb:  TYPE 0,71  ; Left + Middle           = Home      key
mrb:  TYPE 0,79  ;        Middle + Right   = End       key
lrb:  TYPE 0,15  ; Left          + Right   = Shift-Tab key combination
allb: TYPE 5     ; Left + Middle + Right   = Ctrl-E    key combination
```

Then, compile this menu source file (C:\LTM\ICE.DEF) to create the menu definition file (C:\LTM\ICE.MNU):

```
CD C:\LTM
NEWMENU ICE
```

The following is a fragment of an AUTOEXEC.BAT file which automatically loads the mouse driver and installs the iceMASTER mouse menu (C:\LTM\ICE.MNU) created above:

```
REM AUTOEXEC.BAT Fragment for LOGITECH Mouse:
REM
REM (Tested using Version 3.42 of the LOGITECH Mouse Driver)
REM (LOGITECH Mouse software files at "C:\LTM\")
REM
REM 1.  Load Mouse Driver:
REM     (use 'C:\LTM\MOUSE'   if the mouse is on COM1)
REM     (use 'C:\LTM\MOUSE 2' if the mouse is on COM2)
C:\LTM\MOUSE 2
REM
REM 2.  Install iceMASTER Menu.
REM     iceMASTER menu files are  C:\LTM\ICE.DEF
REM                           and C:\LTM\ICE.MNU
REM     This menu defines the mouse buttons as follows:
REM        Movement:
REM          Left  = Left  arrow key
REM          Right = Right arrow key
REM          Up    = Up    arrow key
REM          Down  = Down  arrow key
REM        Buttons:
REM          Left                  = Enter     key
REM                 Middle         = Tab       key
REM                          Right = Esc       key
REM          Left + Middle         = Home      key
REM                 Middle + Right = End       key
REM          Left          + Right = Shift-Tab key combination
REM          Left + Middle + Right = Ctrl-E    key combination
REM
REM     C:\LTM\ICE.DEF is the raw source for the button/movement
REM     definitions.  After creating C:\LTM\ICE.DEF, type
REM         CD C:\LTM
REM         NEWMENU ICE
REM     to create the C:\LTM\ICE.MNU menu definition file.
REM
C:\LTM\MENU  C:\LTM\ICE
C:\LTM\CLICK
REM
REM Later, to dynamically remove one or more of the mouse utilities:
REM    CLICK OFF                          {remove Click       }
REM    MENU OFF                           {remove Mouse Menus }
REM    MOUSE OFF                          {remove Mouse Driver}
```

Mouse Systems White Mouse Support

The following paragraphs show how to set up a mouse menu for iceMASTER using a Mouse Systems White Mouse (3-button).

Assuming that the mouse driver files are located at C:\MOUSE, the following steps will create a mouse menu which can be loaded to control iceMASTER.

You first need to create a menu source file, M_IM.MSC, containing the definitions mapping mouse movements/buttons to keyboard keystrokes. You can use any text editor or word processor to create a plain ("non-document") ASCII text file containing:

```
;
;Mouse Systems White Mouse Configuration for iceMASTER
;
;Cursor Definitions
;
Cursdef: Cursor
(
Left([Left])
Right([Right])
Up([Up])
Down([Down])
Sensitivity(50,35)
Hysteresis(1,1)
)
;
;
;Button Definitions
;
LB:  Button (Keys([Enter]))  ;Left          button
MB:  Button (Keys([Esc]))    ;      Middle         button
RB:  Button (Keys([Tab]))    ;               Right button

LM:  Button (Keys([Home]))   ;Left + Middle         button
MR:  Button (Keys([End]))    ;      Middle + Right button
LR:  Button (Keys([&H0F00])) ;Left          + Right button (Shift-Tab)
LMR: Button (Keys([c-E]))    ;Left + Middle + Right button (Ctrl-E)

;
;Mouse Definition
;
Mouse
(
Left (LB)
Middle (MB)
Right (RB)
LeftMiddle (LM)
MiddleRight (MR)
LeftRight (LR)
LeftMiddleRight (LMR)
Cursor (Cursdef) )
```

Then, compile this menu source file (C:\MOUSE\M_IM.MSC) to create the menu definition file (C:\MOUSE\M_IM_COM):

```
CD C:\MOUSE
MSC M_IM.MSC
```

The following is a fragment of an AUTOEXEC.BAT file which automatically loads the mouse driver and installs the iceMASTER mouse menu (C:\MOUSE\M_IM.COM) created above:

```
REM AUTOEXEC.BAT Fragment for Mouse Systems White Mouse:
REM
REM (Tested using Version 6.23 of the  Mouse Driver)
REM (Mouse Systems software files at "C:\MOUSE")
REM
REM 1.  Load Mouse Driver:
REM     (use 'C:\MOUSE\MSCMOUSE    /A0' 'unaccelerated' mouse on COM1)
REM     (use 'C:\MOUSE\MSCMOUSE /2 /A0' 'unaccelerated' mouse on COM2)
C:\MOUSE\MSCMOUSE /2 /A0
REM
REM 2.  Install iceMASTER Menu.
REM     iceMASTER menu files are  C:\MOUSE\M_IM.COM
REM     This menu defines the mouse buttons as follows:
REM        Movement:
REM          Left  = Left  arrow key
REM          Right = Right arrow key
REM          Up    = Up    arrow key
REM          Down  = Down  arrow key
REM        Buttons:
REM          Left                  = Enter     key
REM                 Middle         = Tab       key
REM                          Right = Esc       key
REM          Left + Middle         = Home      key
REM                 Middle + Right = End       key
REM          Left          + Right = Shift-Tab key combination
REM          Left + Middle + Right = Ctrl-E    key combination
REM
REM     C:\MOUSE\MOUSE.MSC is the raw source for the button/movement
REM     definitions.  After creating C:\MOUSE\M_IM.MSC, type
REM        CD C:\MOUSE
REM        MSC M_IM.MSC
REM     to create the C:\MOUSE\M_IM.COM menu definition file.
REM
C:\MOUSE\POPUP /M:142
C:\MOUSE\M_IM
REM
REM Later, to unload the mouse driver and menu:
REM    C:\MOUSE\MSCMOUSE /U
```

Appendix J: Command Line Options

The emulator Host Software command line has the following form:

ICE [option(s)]

The command line options may be specified in any order and they are case-sensitive.

To load a file into the emulator without having to configure the emulator and load a program as separate steps, simply include the name of the file you wish to load on the command line. For example:

ICE F_DEMO.AOM

will configure (*Configure*|*Emulator*|*Execute*, page 7-2) the emulator and load (*File*|*Load*, page 7-27) this file for immediate use.

Other command line options include:

-m {FILENAME.EXT}	Use {FILENAME.EXT} as model file instead of $MODEL.
-i {FILENAME.EXT}	Use {FILENAME.EXT} as macro file to execute. This macro input file must have been created during a previous debug session.
-s {path}	Use {path} to locate High-Level Language (HLL) source files. If not specified
-*l*	Ignore local symbol debug records when loading a program file (absolute object module / load module). Debug records for global (public) symbols are still processed and the global symbols can be referenced symbolically from within the Host Software. This option is useful when the Host Software runs out of memory when loading a very large program with an extremely large number of debug records for local and global symbols.
-v	Disable testing (loading) of all possible video character sets (screen sizes) during Host Software initialization. The user must ensure that the PC is in some usable video mode when the Host Software is invoked, such as BIOS Video Mode 3 (default for CGA, EGA, VGA). This option is provided for some video boards which evidently are not 100% Video BIOS compatible.
-b< num >	In the absence of a $CONFIG file, set the default initial baud rate as follows: -b0 115200 {default} -b1 57600 -b2 38400 -b3 28800 -b4 19200 -b5 9600

-hxyz

This option gives users some control over the mechanisms used by the Host Software in determining the beginning and ending addresses for each module in a program. This information is not recorded explicitly in an AOMF (Absolute Object Module Format) file, but must be derived by the Host Software when the program is loaded into the emulator. The Host Software, using several different kinds of information present in the AOMF file, attempts to do the best job of determining the module boundaries. However, due to movement of code segments/sections by the linker, or the presence of absolutely ORG'ed segments in an otherwise relocatable module, these attempts can cause the Host Software to "shorten" or "lengthen" or "lose" modules in the user's program. Note that when this happens, the ONLY side effects are that source line images may not be found for a particular module (due to truncation of the debug line number table) or the 'current module' (reported in various contexts) may not be correct. The actual code stream image seen and processed by the emulator is correct and all symbols, both global and local, in the AOMF file are available to the Host Software and have the correct addresses associated with them. Note that 'x', 'y' and 'z' may be either '0' (don't use) or '1' (use), as follows:

x	Use min/max code text initialization addresses from each module.
y	Use min/max source line number addresses from each module.
z	Use min/max code label (local or global) addresses from each module.

The default is '-h111': Use all three pieces of information in determining the minimum and maximum addresses for each module. You may see better results for some programs with:

h010 (Use only min/max source line number addresses in determining the beginning and ending addresses for each module).

or

h110 (Use both min/max code label addresses and min/max source line number addresses in determining the beginning and ending addresses for each module).

-d < ms >

Set the macro playback delay between keystrokes to < ms > milliseconds. To be used in conjunction with the '-i' command line argument.

-n

Forces interpretation of the 'blink' video attribute bit to mean "intensify the background color". Effectively, this allows use of 16 background colors rather than the normal 8. This option should be specified only when the PC has an EGA/VGA adapter installed, attached to an EGA/VGA monitor. Normally, the Host Software automatically determines how to interpret the 'blink' bit, based on the adapter and monitor installed in the system. However, sometimes the Software can get "fooled". This option provides an absolute override.

-o	Forces interpretation of the 'blink' video attribute bit to mean "blink the foreground character". This option should be specified when the PC has a Monochrome or CGA adapter installed, attached to a monochrome or CGA monitor. Normally, the Host Software automatically determines how to interpret the 'blink' bit, based on the adapter and monitor installed in the system. However, sometimes the Software can get "fooled". This option provides an absolute override.
-q	Create a command chart (included in Appendix O, the Command Chart). The command chart is written to the file CRQ_51.TXT.
-qv	Create a "verbose" command and Help Topic reference. The output is written to the file CRQV_51.TXT.

When running under DOS, the forward slash, '/ ', may also be used as the option/argument indicator (instead of the dash, '-').

Appendix K: Host Software Files

iceMASTER Host Software System Files

The Host Software executable file is:

ICE.EXE

The ICE.EXE file is the core of the Host Software system. Executing it brings the user into the Host Software environment. The ICE.EXE file is distributed on the distribution diskette labeled Host Software.

The following files are read and/or written by the Host Software in order to configure various parts of the Host Software environment. When the Host Software attempts to open these files, it searches the following locations, in this order:

1) The current directory.
2) The paths you have defined within the DOS APPEND statement.
3) The path specified by the EHSFP (Emulator Host Software File Path) environment variable (see AUTOEXEC.BAT in DOS Information appendix).

A short description of each system file follows:

$ALERT Optional File: Yes

Usage: The $ALERT file contains the Alert options settings specified in the *Configure|Options* screen (page 7-21). The $ALERT file is read during Host Software initialization and when the *Configure|Restore* command (page 7-26) is selected.

How Created: The $ALERT file is created, or rewritten, whenever the *Configure|Save|Save* command (page 7-25) is selected. The $ALERT file is a binary file and must not be modified.

$CLR1$ Optional File: Yes

Usage: The $CLR1$ file contains the default color and video attribute settings suitable for CGA monitors. The $CLR1$ file is read by the Host Software during video initialization if there is no $COLOR file and the monitor type is a CGA. The $CLR1$ file is read unconditionally if the *Configure|Attributes|CGA/EGA/VGA Default* command (page 7-8) is selected.

How Created: The $CLR1$ file is supplied as part of the Host Software system and will never be written to by the Host Software. The $CLR1$ file is a binary file and must not be modified.

$CLR2$ Optional File: Yes

Usage: The $CLR2$ file contains the default color and video attribute settings suitable for EGA/VGA monitors. The $CLR2$ file is read by the Host Software during video initialization if there is no $COLOR file and the monitor type is an EGA or VGA. The $CLR2$ file is read unconditionally if the *Configure|Attributes|EGA/VGA Default* command (page 7-8) is selected.

How Created: The $CLR2$ file is supplied as part of the Host Software system and will never be written to by the Host Software. The $CLR2$ file is a binary file and must not be modified.

$CLRM$ Optional File: Yes

Usage: The $CLRM$ file contains the default color and video attribute settings suitable for Monochrome monitors. The $CLRM$ file is read by the Host Software during video initialization if there is no $COLOR file and the monitor type is a Monochrome. The $CLRM$ file is read unconditionally if the *Configure|Attributes|Monochrome Default* command (page 7-8) is selected.

How Created: The $CLRM$ file is supplied as part of the Host Software system and will never be written to by the Host Software. The $CLR1$ file is a binary file and must not be modified.

$CODMEM$ Optional File: Yes

Usage: The $CODMEM$ file is a temporary (scratch) file which contains the raw, binary image of code memory currently in the emulator. It is used to minimize memory consumption within the Host Computer and to speed processing during complex break evaluation.

How Created: The $CODMEM$ file is created during Host Software initialization and is read and written during many Host Software operations.

$COLOR Optional File: Yes

If the $COLOR file does not exist, the Host Software will use default colors and video attributes, based on the monitor type (monochrome or color).

Usage: The $COLOR file is read by the Host Software during its one-time initialization sequence, or when you execute the *Configure|Restore* command (page 7-26) or *Configure|Attributes|User* command (page 7-9). The $COLOR file contains the user-specified color and video attribute settings to be used by the Host Software.

How Created: The $COLOR file is created, or rewritten, only when the user saves the current color and video attribute settings with the *Configure|Save|Save* command (page 7-25). The $COLOR file is a binary file and must not be edited directly.

$CONFIG Optional File: Yes

If the $CONFIG file does not exist, the Host Software will use default configuration settings when establishing communication with the emulator base.

Usage: The $CONFIG file is read by the Host Software during its one-time initialization sequence, and whenever the emulator must be configured. The $CONFIG file contains an encoding of:

1) the baud rate to be used in communication between the Host Software and the emulator base (see *Configure|Emulator|Baud rate*, page 7-3)
2) the RS-232 communication port to be used in communication between the Host Software and the emulator base (see *Configure|Emulator|Comm port*, page 7-3)
3) the current "chip mode of operation" (see *Configure|Emulator|Mode*, page 7-2)

How Created: The $CONFIG file is created, or rewritten, whenever any of the following commands are selected:

Configure|Emulator|Mode

Configure|Emulator|Baud rate

Configure|Emulator|Comm port

The $CONFIG file is a binary file and must not be edited directly.

$FKEYDEF Optional File: Yes

If the $FKEYDEF file does not exist, the Host Software will use default Function Key (Hot Key) settings and will display the default number (2) of Function Key label lines at the bottom of the screen.

Usage: The $FKEYDEF file is read by the Host Software during its one-time initialization sequence, or when the *Configure|Restore* command (page 7-26) is selected. The $FKEYDEF file contains an encoding of:

1) the 40 Function Key assignments

2) the number of Function Key label lines displayed at the bottom of the screen.

How Created: The $FKEYDEF file is created, or rewritten, whenever the *Configure|Save|Save* command (page 7-25) is issued. The $FKEYDEF file is a binary file and must not be edited directly.

$LINE Optional File: Yes

If the $LINE file does not exist, the Host Software will use the default screen size.

Usage: The $LINE file is read by the Host Software during its one-time initialization sequence. The $LINE file contains an encoding of the screen size (video mode).

How Created: The $LINE file is created, or rewritten, whenever the *Configure|Save|Save* command (page 7-25) is issued. The $LINE file is a binary file and must not be edited directly.

$MISC Optional File: Yes

Usage: The $MISC file contains the Miscellaneous options settings specified in the *Configure|Options* Pull-down Menu (page 7-21). The $MISC file is read during Host Software initialization and when the *Configure|Restore* command (page 7-26) is selected.

How Created: The $MISC file is created, or rewritten, whenever the *Configure|Save|Save* command (page 7-25) is executed. The $MISC file is a binary file and must not be modified.

$MODEL Optional File: No

Usage: The $MODEL file is read by the Host Software during its one-time initialization sequence. The $MODEL file defines the operational properties of the particular probe card being used. The $MODEL file also specifies the settings for several user-configurable aspects of the Host Software's operating environment. The $MODEL file also contains the Help information.

How Created: The $MODEL file is created by using the MF_GEN.EXE utility program distributed with the files on the Probe Card Software distribution diskette. After creation, the $MODEL file may be modified by using any text editor. The $MODEL file is a simple ASCII text file. See Appendix M, Model File Configuration for a description of those directives (lines) in the $MODEL file which may be modified.

$TRDAT$ Optional File: Yes

Usage: The $TRDAT$ file is a temporary (scratch) file which contains the raw, binary image of the current trace buffer. It is used to minimize memory consumption within the Host Computer and to speed processing during trace buffer decoding.

How Created: The $TRDAT$ file is created during Host Software initialization.

$WINDOW Optional File: Yes

Usage: The $WINDOW file contains the current window definitions (size and position) of the Main Screen windows and other windows capable of being sized or moved. The $WINDOW file is read during Host Software initialization and when the *Configure | Restore* command (page 7-26) is selected.

How Created: The $WINDOW file is created, or rewritten, whenever the *Configure | Save | Save* command (page 7-25) is selected. The $WINDOW file is a binary file and must not be modified.

Host Software File Organization

If you have more than one emulator, consider making separate and easily recognized directories for each emulator. For example, if you have both an iceMASTER emulator and a MicroICE + emulator, you might lay out the directory structure as follows:

```
                  ┌──── (other directories)
                  │
ROOT ─────────────┼──── IM51 (for iceMASTER)
                  │
                  └──── MIP51 (for MicroICE+)
```

In any case, it is important that you

Do Not Mix Files From Different Emulators

Appendix L: DOS Information

Host Computer Environment Settings

In order for the Host Computer to properly execute the Host Software system commands, the user must configure the DOS environment properly. The important files are:

CONFIG.SYS

The important parameters for the file CONFIG.SYS are the number of files that can be open at any one time (FILES = XX) and the number of buffers available to the system (BUFFERS = XX). Setting both of these parameters to at least 20 is acceptable for iceMASTER unless there are other applications that recommend a setting of more that 20. The CONFIG.SYS file may also have other specifications for the Host Computer environment, including SHELL, DEVICE, and BREAK. Consult the DOS reference manual (as well as the manuals for other applications that will be running on the Host Computer) for details on these parameters.

A typical DOS 4.01 CONFIG.SYS file might read:

```
REM Example CONFIG.SYS file for DOS Version 4.01:
BREAK=ON
BUFFERS=20
FILES=20
REM Increase environment size to 2048 (2K) bytes:
SHELL=C:\DOS401\COMMAND.COM /P /MSG /E:2048
REM Use the DOS 4. 01 ANSI.SYS Display Device Driver:
DEVICE=C:\DOS401\ANSI.SYS /X
```

A typical DOS 3.30 CONFIG.SYS file might be:

```
REM Example CONFIG.SYS file for DOS Version 3.30:
BREAK=ON
BUFFERS=20
FILES=20
REM Increase environment size to 1048 (2K) bytes:
Shell=C:\DOS330\COMMAND.COM /P /E:2048
```

COMMAND.COM

Several emulator Host Software commands (such as *File | OS Escape*) work by making DOS calls. In order to do this, a command processor must be available. The default command processor for DOS is COMMAND.COM. Another command processor can be specified using the SHELL command within CONFIG.SYS. Normally, the command processor resides in the root directory of the default drive, but if not the COMSPEC environment variable mus be set to point to the command processor.

Note that changing the COMSPEC environment setting from its default setting may cause problems if not done properly. Consult the DOS Reference Manual for further information on the COMSPEC environment variable.

AUTOEXEC.BAT

The AUTOEXEC.BAT file is automatically executed by DOS each time the Host Computer is powered up or re-booted.

Typically the PATH and EHSFP environment variables should be initialized in this file. A typical AUTOEXEC.BAT file, with comments, follows:

```
REM Example AUTOEXEC.BAT file for DOS Version 4.01 when booting from
hard drive 'C:':
ECHO  OFF
REM ---------------------- Directory Information --------------------
-
REM C:\C51V2 Archimedes C-51 Version 2.0
REM C:\DOS401 MS-DOS 4.01 files
REM C:\F_C51 Franklin Software's 8051 C/Asm/Linker (newest version)
REM C:\ICE MetaLink Host Software and associated files
REM C:\IS Intel ASM-51, PL/M-51, RL51
REM C:\MCCC Micro Computer Control Corp. 'mcc51' (MICRO/C-51)
REM C:\MISC Miscellaneous Utilities/Files/Programs
PATH C:\DOS401;C:\F_C51;C:\IS;C:\C51V2;C:\MCCC;\MISC;C:\ICE;C:\
REM EHSFP Emulator Host Software File Path
SET   EHSFP=C:\ICE\
REM C_INCLUDE is for Archimedes 8051 C
SET  C_INCLUDE=C:\C51V2\
REM C51LIB is for Franklin Software 8051 C
SET  C51LIB=C:\F_C51
REM :INCLUDE: is for Franklin Software 8051 C
SET  INCLUDE:=C:\F_C51
REM TMP is for Franklin Software 8051 C
SET  TMP=C:\C\TMP
REM The COMSPEC variable specifies the location of thecommand processor
REM (The default location, when booting from hard drive C:, is C:\COM-
MAND.COM):
COMSPEC=C:\DOS401\COMMAND.COM
REM ANSI.SYS in DOS 4.0 now supports other than 25 line displays:
MODE CON: LINES=43
REM Display DOS Version Number:
VER
REM Analyze disk directory and file allocation table for consistency
and report any errors:
CHKDSK
REM Display current date and time without requiring user intervention:
PROMPT $d $t $g @echo off
REM Using ANSI.SYS commands, specify the screen colors and the format
of the DOS prompt:
REM 1) Display the DOS prompt in reverse video (black on white)
REM   $e[7m
REM 2) Define the DOS prompt as the current drive and path,followed by
a greater-than sign:
REM $p$g ($p = current drive and path, $g = character)
REM 3) Set normal screen colors to white on a blue background:
REM $e[37;44m(37 = white foreground, 44 = blue background)
@ECHO ON
PROMPT $e[7m$p$g$e[37;44m
```

You could use this AUTOEXEC.BAT file in DOS 3.30 as well, if the Mode command (DOS versions prior to 4.01 won't recognize the 'MODE CON: LINES: =43' statement) is removed. If you have named your DOS directory something besides "DOS401", be sure to use the path information that is valid for your drive. Refer to your DOS Reference Manual for more information.

Editing DOS Files

To build the CONFIG.SYS and AUTOEXEC.BAT files, there are several text editing methods. In general, any editor or word processor capable of editing a simple ASCII text (non-document) file can do the job. In addition, the DOS COPY command can use the DOS device CON: to build short files. The syntax is:

```
COPY CON: FILENAM.EXT
statement 1
statement n
^Z
```

Therefore, the file CONFIG.SYS could be created by the following text, typed from the DOS command line:

```
COPY CON: CONFIG.SYS
FILES=20
BUFFERS=20
^Z
```

Appendix M: Model File Configuration

$MODEL File Overview

The $MODEL file is a plain ASCII text file which may be edited using any text editor. The $MODEL file consists of a series of directives which control the behavior of the Host Software and the emulator. Only those directives listed in this appendix may be modified by you. All other directives in the $MODEL file must remain unchanged.

WARNING: Any change to the $MODEL file other than those discussed here may result in erratic operation, or no operation at all.

If you wish to change any of the following directives, we recommend that you first make a backup copy of the original $MODEL file. A new $MODEL file may be generated using the MF_GEN.EXE utility provided with the Probe Card Software distribution diskette.

A11: SFR Display Sort Order

This directive specifies the order in which the SFRs (Special Function Registers) will be displayed in the Register Window. The format of the A11 directive is:

A11 {SFR_disp_sort_order};

where

{SFR_disp_sort_order} can be:

- 0 display SFRs (left-to-right, top-to-bottom) in the order supplied in the Model File
- 1 display SFRs (left-to-right, top-to-bottom) alphabetically (ASCII collating sequence)
- 2 display SFRs (left-to-right, top-to-bottom) in increasing address order

The default A11 directive is :

A11 0; #display SFRs (left-to-right, top-to-bottom) in the order supplied in the Model File

A36: Informative Messages During Program Load

The format of the A36 directive is:

A36 {post-file-delay}{post-module-delay};

where

{post-file-delay} is the number of milliseconds to delay after (re)establishing the correspondence between entries in the Line Number Table and source images (file seek positions) for each module in the file currently loaded into code memory. This delay occurs after each

Source Found For Name

message is displayed. This allows the messages (if any) regarding the Host Software's ability to locate source images to remain on the screen long enough for you to read it. A value of 0 indicates no delay. The maximum allowed value is 30000 (30 seconds).

{post-module-delay} is the number of milliseconds to delay after (re)establishing the correspondence between entries in the Line Number Table and source images (file seek positions) for the last module in the file currently loaded into code memory. This additional delay occurs after the last message is displayed, or after the

No Source Line Numbers Present

or

Source Line Numbers Present, but No Source Files Found

message is displayed. This is done to allow the last message (if any) regarding the Host Software's ability to locate source images to remain on the screen long enough for you to read it. A value of 0 indicates no delay. The maximum allowed value is 30000 (30 seconds).

If both values are specified as zero (0), no messages are ever displayed and there is no delay.

The default A36 directive is:

A36 0;	#Post-file delay = 0 milliseconds
0;	#Post-module delay = 0 milliseconds

A40: Specify Method Of Updating (Writing To) Video Display

The format of the A40 directive is:

A40 {technique} {video_RAM_address_override} {MBZ};

where

{technique} can be:

0 (DEFAULT) Perform PC video display output by writing directly to video RAM. This option should be specified only when using PC's that are 100% IBM hardware compatible. This is the fastest technique.

If you have an older CGA video controller, you may notice some flickering in the display when this technique is used. The flickering can be eliminated by choosing Technique 1. The ANSI.SYS device driver may or may not be loaded, at your discretion, with no ill effects. For more information refer to {video_RAM_address_override} below.

1 Perform PC video display output by using BIOS interrupts (only). This option should be specified when using PC's that are not 100% IBM hardware compatible, but that do have a 100% IBM compatible BIOS. This is the next fastest technique. The ANSI.SYS device driver may or may not be loaded, at your discretion, with no ill effects.

2 Perform PC video display output by using (indirectly) DOS interrupts, via the internal C run-time library. This option should be specified when using PC's that are not 100% IBM hardware compatible and do not have a 100% IBM compatible BIOS. This is the slowest technique.

When this technique is used with a color monitor and the device driver ANSI.SYS has been loaded, the colors in effect at the time the Host Software is invoked will "bleed through" into the display.

{video_RAM_address_override}

The value in this field has meaning only when the video display {technique} above is 0 (write directly to video RAM) and the value specified here is non zero. The Host Software automatically uses the following base addresses for video RAM:

0xB000 for a Monochrome Display Adapter

0xB800 for a Color/EGA Display Adapter

If your display adapter is set up to use a different address, then specify that address here. It should be in the form 0xhhhh where "hhhh" is the hexadecimal address of the base of the video RAM segment.

{MBZ}

0 Must Be Zero; reserved for future use.

The default A40 directive is:

A40 0 0x0000 0;#display video output by writing directly to video RAM

A41: Control Formatting Of HLL Source Images

The A41 directive specifies how HLL source images will be formatted for display in all contexts in the Host Software. The format of the A41 directive is:

A41 {skip_leading_blanks} {tab_stop_setting};

where

{skip_leading_blanks}

0 (DEFAULT) Do not skip (ignore) leading blanks when displaying HLL source images.

1 Skip (ignore) leading blanks when displaying HLL source images.

{tab_stop_setting}

The tab stop value in this field can range from 0 to 10. It specifies the tab stop setting (column width) to be used in processing ASCII horizontal tab characters (09 hexadecimal) in the HLL source images. The default value is 4. A value of 0 means that ASCII horizontal tab characters will be ignored. A value of 1 means that each ASCII horizontal tab character will be replaced with a single blank.

Note that for Intel PL/M-51 the {tab_stop_setting} value does not really apply to PL/M-51 modules. The PL/M-51 compiler expands all ASCII tab characters present in the source file into the appropriate number of blanks when generating the program listing (.LST) file. The Host Software obtains PL/M-51 source images from this '.LST' file. Therefore, the only way to retain the indentation reflected in the .LST file is to specify a {skip_leading_blanks} value of 0 (which is the default meaning don't skip leading blanks).

The default A41 directive is:

A41 0; #HLL source display: leading blanks skip (1)/don't skip (0)
4; #HLL source display: tab stops (column width) for ASCII TAB chars(0-10)

A42: Performance Analyzer Display Characters

The A42 directive allows you to specify certain special characters used by the Performance Analyzer and to control other aspects of the Performance Analyzer's operation. The format of the A42 directive is:

A42 {bar_blot_char} {file_bar_blot_char}
{bar_half_blot_char} {file_bar_half_blot_char}
{micro_char} {file_micro_char}
{vert_bar_char} {file_vert_bar_char}
{display mode};

The default A42 directive is:

A42 0xDB	# 4% bar char (video display)
'#'	# 4% bar char (when writing to file)
0x10	# 2% bar char (video display)
'> >'	# 2% bar char (when writing to file)
0xE6	# "micro-" (mu) symbol (video display)
'u'	# "micro-" (mu) symbol (when writing to file)
0xB3	# vertical line char (video display)
'|'	# vertical line char (when writing to file)
0;	# in Symbolic and Mixed mode, display both Global & Local code labels
	# (0 = Globals & Locals, 1 = Globals only, 2 = Locals only)

A43: Three Digit Separator

The character specified here will be used as the separator between every third digit when displaying the real-time clock value in Main Status Window, the Performance Analyzer Emulation Window, and sample counts in the Performance Analyzer Emulation Window.

The default A43 directive is:

A43 ',';   # Separator for every 3rd digit in time/count values. #Note, the Europeans separator is a period.

A49: RS232 Time-out Loop Count

The format of the A49 directive is:

A49 {normal time-out count} {short time-out count};

where

{normal time-out count}

sets the time-out limit for normal reception from, and transmission to, the emulator. Note that the number specified here is not an actual clock time value, but rather a count for the number of times the Host Software will execute (wait in) the innermost "wait for a byte to be received" or "wait for a byte to be transmitted" loop. The larger the number, the longer the time-out interval will be (i. e., the more times the Host Software will execute the inner most "wait" loop before declaring an actual time-out). Generally, faster Host Computers will require a larger number here; slower Host Computers can get by with a smaller number. Note that if the number here is too large and there actually is a communication failure problem, it will take a very long time for the Host Software to detect and report the problem.

{Short time-out count}

sets the time-out limit for the quick communication test routine which is used to detect a warm-start condition when the Host Software is first invoked. The larger the number specified here, the longer the Host Software will wait for a response from the emulator during the Host Software's one-time-initialization (only performed when the Host Software is first invoked).

The default A49 directive is:

A49 5	#1st RS232 communication time-out limit (normal comm)
4;	#2nd RS232 communication time-out limit (warm-start comm test)

A50: Data Type Display Modes

The A50 directive is used to set the display mode for each data type when displaying HLL symbol values. Currently, the Franklin and Keil C compilers output C data information when the OBJECTEXTEND compilation option is activated. The format of the A50 directive is:

A50 {char_dsply_mode} {uchar_dsply_mode}
{int_dsply_mode} {uint_dsply_mode}
{long_dsply_mode} {ulong_dsply_mode}
{float_dsply_mode};

where

0	hex format
1	decimal format
2	character format

The default A50 directive is:

A50 0	# char default: hex
0	# unsigned char default: hex
1	# int default: decimal
1	# unsigned int default: decimal
1	# long default decimal
1	# unsigned long default: decimal
0;	# float default: hex

Appendix N: Default Function Key Assignments

Function Key	Short Description	Command Sequence
F1	Help	Help
F2	ResetEm	*Run|Reset|Emulator*
F3	ResetTg	*Run|Reset|Target*
F4	Go	*Run|Go*
F5	GoFrom	*Run|From*
F6	GoUntil	*Run|Until*
F7	StepIns	*Run|Step*
F8	StepLin	*Run|Line*
F9	StepOvr	*Run|Over*
F10	StepTo	*Run|To*
Shift-F1	Load	*File|Load*
Shift-F2	SetBkpt	*Break/Trace|Set*
Shift-F3	ViewTrc	*Break/Trace|View Trace*
Shift-F4	DisAssm	*Display/Alter|Asm/Dasm*
Shift-F5	MacExec	*File|Macro|Execute*
Shift-F6	CurMod	*Source/Symbols|Current Module*
Shift-F7	SrcPath	*Source/Symbols|Source Path*
Shift-F8	RawSrc	*Source/Symbols|Raw Source*
Shift-F9	SymGlob	*Source/Symbols|Global*
Shift-F10	SymAlph	*Source/Symbols|Alpha*
Ctrl-F1	Modules	*Source/Symbols|Modules*
Ctrl-F2	Scopes	*Source/Symbols|Scopes*
Ctrl-F3	LinNums	*Source/Symbols|Line Numbers*
Ctrl-F4	SymLocl	*Source/Symbols|Local*
Ctrl-F5	SymAddr	*Source/Symbols|Address*
Ctrl-F6	WnModfy	*Configure|Windows|Modify*
Ctrl-F7	WnResiz	*Configure|Windows|Size*
Ctrl-F8	WnSelct	*Configure|Windows|Goto*
Ctrl-F9	AddWtch	*Configure|Windows|Add*
Ctrl-F10	DelWtch	*Configure|Windows|Delete*
Alt-F1	WnRepnt	*Configure|Windows|Repaint*
Alt-F2	MemCode	*Display/Alter|Code*
Alt-F3	MemIdat	*Display/Alter|Idata*
Alt-F4	MemXdat	*Display/Alter|Xdata*
Alt-F5	SymVwCh	*Display/Alter|Var/Reg*
Alt-F6	RAM Bit	*Display/Alter|RAM-Bits*
Alt-F7	GoSlow	*Run|Slow Motion*
Alt-F8	RepCnt	*Run|Repetition Count*
Alt-F9	HRes PA	*Misc|High Resolution*
Alt-F10	HBin PA	*Misc|High Bin Count*

Appendix O: iceMASTER Command Chart

A command chart is listed on the following pages. The command chart includes (in column order):

1) Menu/Command Tree Structure
2) Val - Default value (if applicable)
3) H-Key - Default Hot Key (Function Key) assignment
4) Lab - Short command description used for Function Key Labels at bottom of screen ('...' if not assignable to a Function Key)
5) Menu/Command Description (copy of "Quick Help" line)

Note that this command chart can be generated by invoking the Host Software using the '-q' command line option. If you reassign any Function Keys or change any default values, use the '-q' command line option to generate a new chart and replace the chart on the following pages.

Menu/Command Tree Structure	Val(1)	H-Key(2)	Lab(3)	Menu/Command Description (copy of "Quick Help" line at bottom of screen)
Configure				Set up the system configuration
Emulator				Configure the emulator
Execute				Configure system according to current settings (shown here & saved in $CONFIG)
Mode	Mode 1			Select probe card mode of operation (ROM, ROMless, etc.)
Mode 1: ROMless (803X) Operation, /EA = Low				
Mode 2: ROM (805X), /EA = High				
Comm port	COM1			Select RS-232 communication port
COM1				Use COM1 for communication between host PC and emulator base
COM2				Use COM2 for communication between host PC and emulator base
Baud rate	115200			Select RS-232 communication baud rate (serial link speed)
115200				
57600				
38400				
28800				
19200				
9600				
(Mapping)				
Code Memory				Display/change Code memory mapping
Emulator				Map a block of code memory to the emulator
Target				Map a block of code memory to the target board
All				Map all code memory menu
Emulator				Map all code memory to the emulator
Target				Map all code memory to the target board
Load				Load map configuration from a file
Save				Save current map configuration to a file
Xdata Memory				Display/change External Data memory mapping
Emulator				Map a block of external data memory to the emulator
Target				Map a block of external data memory to the target board
All				Map all external data memory menu
Emulator				Map all external data memory to the emulator
Target				Map all external data memory to the target board
Load				Load map configuration from a file
Save				Save current map configuration to a file
(Display)				
Attributes				Change video display parameters (e.g., colors and number of lines on screen)
Lines	25			Set size of video display (number of lines)
25				25-line display
28				28-line display
43				43-line display
50				50-line display
(Colors)				
CGA/EGA/VGA Default				Set colors to supplied defaults suitable for CGA/EGA/VGA monitors (file $CLR1$)
EGA/VGA Default				Set colors to supplied defaults suitable for EGA/VGA monitors (file $CLR2$)
Monochrome Default				Set colors to supplied defaults suitable for monochrome monitors (file $CLRM$)
User				Set video display colors to your default colors (file $COLOR)
Select				Select all video display colors and attributes individually
Windows				Change layout/size of main windows, "goto" a window & add/delete watches

Modify		Ctrl-F6	WnModfy	Modify individual window attributes (active, order, start address, misc options)
Size		Ctrl-F7	WnResiz	Manually adjust the size of one or more windows on the main screen
Goto		Ctrl-F8	WnSelct	"Jump into" (select) a main screen window, to peruse and/or change data in it
Repaint		Alt-F1	WnRepnt	Repaint all windows on the main screen (e.g., to see changing register values)
(Watch)				
Add		Ctrl-F9	AddWtch	Add a watch variable to the Watch Window
Delete		Ctrl-F10	DelWtch	Delete (remove) a watch variable from the Watch Window
Identification				Display software and hardware version/environment information
Options				View/change miscellaneous user interface configuration options
(Alert)				
Diagnostics	On			Beep when a diagnostic message is issued
Bad key/command	On			Beep when an unrecognized key is pressed
Host-break	On			Beep at a Host-Break (i.e., when Esc key is pressed to halt emulation)
(Miscellaneous)				
Flicker	On			Flicker response in "Yes/No" dialog boxes, etc.
Trace prefetch	On			Automatically prefetch trace frames from emulator after each emulation cycle
Main Esc	On			Pressing Esc key in Main Menu will prompt to exit Host Software
Common Window Borders	On			Adjacent Main Screen Windows share a common bottom/top border
Stack Hyperlinks	On			Hyperlinks in 'Help' stack, allowing return to previous hyperlink topic
Prompt OS-Escape	On			During OS-Escape, augment DOS prompt with "...type 'exit' to return..." reminder
Unknown data type size	1			Number of bytes to display for unknown data types in 'Source/Symbol' displays
Function-Keys				View/change current Function Key assignments & number of label lines displayed
Lines	2			Specify number of Function Key label lines to be displayed at bottom of screen
Modify				View/change current Function Key assignments
Assign				Assign current option to current Function Key
Clear				Clear assignment of current Function Key
Display				Toggle screen display mode
Save				Save Function Key assignments
Write				Write the Function Key assignments to an ASCII text file
Help				Help for Function Key Assignment Menu
Save				Save (change default value of) one or more user interface configuration options
Save				Save current user interface configuration parameters per 'Save Opts' settings
(Save Opts)				
Alert	On			Save current 'Alert' option settings into $ALERT
Colors	On			Save all current screen colors & other video attribute settings into $COLOR
Function-Keys	On			Save all current Function Key specifications into $FKEYDEF
Lines	On			Save current screen size (number of rows) into $LINE
Misc	On			Save current 'Miscellaneous' option settings into $MISC
Windows	On			Save current window definitions (order, size, etc.) into $WINDOW
Restore				Restore all saved user interface configuration options/parameters
File				File operations
Load		Shift-F1	Load	Load program memory from a disk file (AOMF/.HEX/S-Record)
Store				Store current application environment in a disk file
Restore				Restore application environment from a disk file
Upload				Upload program memory from target system board
Download			DownLod	Download target system board external data memory from a disk file
Macro				Create or execute a macro command file
Execute		Shift-F5	MacExec	Playback commands from a macro disk file (created using 'Learn' command below)
Learn				Enter learn mode, "remembering" keystrokes into a macro disk file
Delay	0			Set delay between keystrokes during macro execution (units: 1/1000th second)
Repeat	0			Set macro execution (playback) repeat count

Menu	Default	Key	Label	Description
OS Escape		Alt-O	OS-Esc	Temporarily escape to DOS -- type 'exit' to resume here
Exit		Alt-X	Exit	Terminate host software (quit; exit to DOS)
Run				Begin emulation
Reset				Reset the MCS-51 processor in the probe card, optionally beginning emulation
Processor			ResetPr	Reset the processor (chip) in the probe card, but do not begin emulation
Emulator		F2	ResetEm	Begin emulation from RESET conditions (RESET signal supplied by emulator base)
Target		F3	ResetTg	Begin emulation from RESET conditions (RESET signal supplied by target system)
(Go)				
Go		F4	Go	Begin (resume) emulation at current PC
From		F5	GoFrom	Begin (resume) emulation, specifying a new PC value
Until		F6	GoUntil	Begin (resume) emulation at current PC, specifying a temporary break-point
Slow Motion		Alt-F7	GoSlow	Begin (resume) emulation at current PC in variable-speed "slow motion" mode
(Step)				
Step		F7	StepIns	Single-step from one machine instruction to the next
Line		F8	StepLin	Single-step by source line number (across all modules in program)
Over		F9	StepOvr	Single-step by source line (within current module only: "step over" calls)
To		F10	StepTo	Single-step to procedure/function entry points (global/public code labels)
Repetition Count	1	Alt-F8	RepCnt	Number of automatic "Go" or "Step" repeats of next Reset-/Go-/Step-type command
Display/Alter				Display/Alter memory
Asm/Dasm		Shift-F4	DisAssm	Patch instructions into Code memory or view memory as disassembled instructions
Disassemble				Disassemble code memory bytes
Assemble				Assemble instructions into code memory (patch code memory with new instructions)
Mode				Toggle display mode between Code (instrs only) and Mixed (instrs + HLL images)
Label-synch				Toggle disassembly label-synchronization on/off
Write				Write disassembled code to a file, using current display mode
(Memory)				
Code		Alt-F2	MemCode	View/change/fill/move/compare/write Code memory in hex & ASCII
Browse				View/Change Code memory
Fill				Fill a block of Code memory
Move				Move (copy) a block of Code memory
Compare				Compare two blocks of Code memory
Write				Write Code memory to disk in various formats
Dump				Write Code memory to a file in 'dump' (human-readable) format
Hex				Write Code memory to a file in Intel Standard Hex format
S-Record				Write Code memory to a file in Motorola S-Record format
Idata		Alt-F3	MemIdat	View/change/fill/move/compare/write Internal Data memory in hex & ASCII
Browse				View/Change Internal Data memory
Fill				Fill a block of Internal Data memory
Move				Move (copy) a block of Internal Data memory
Compare				Compare two blocks of Internal Data memory
Write				Write a block of Internal Data memory to a disk file in human-readable form
Xdata		Alt-F4	MemXdat	View/change/fill/move/compare/write External Data memory in hex & ASCII
Browse				View/Change External Data memory
Fill				Fill a block of External Data memory
Move				Move (copy) a block of External Data memory
Compare				Compare two blocks of External Data memory
Write				Write a block of External Data memory to a disk file in human-readable form
Var/Reg		Alt-F5	SymVwCh	View/change, by name, the value assigned to any variable, register or bit
RAM-Bits		Alt-F6	RAM Bit	View/change individual bits in the bit-addressable internal Bit Memory space
Misc				Miscellaneous capabilities
EMM Usage				Display usage of Expanded Memory

PROM Programmer				Program and test '751/'752 PROMs
A-to-D Converter				Display values in A-to-D Converter registers
Multiply-Divide Unit				Display values in Multiply-Divide Unit (MDU) registers (as int or unsigned int)
DPTRs (multiple)				Display values in each of the DPTRn registers
FIFO				Display values in the FIFO
(Performance Analyzer)				
High Resolution		Alt-F9	HRes PA	High resolution performance analysis (program profiling)
Statistics				Performance Analyzer statistics options
Clear				Clear performance analysis statistics
Accumulate	Off			Accumulate performance analysis statistics toggle
View				View performance analysis statistics (post-emulation)
Raw				Display Mode: bar graph lines only (one per bin)
Symbolic				Display Mode: bar graph lines + all labels (global and/or local) in range
HLL				Display Mode: bar graph lines + HLL source images/lines in ranges in bin
Mixed				Display Mode: bar graph lines + labels + HLL source images (Symbolic + HLL)
Counts				Toggle bar graph lines to/from actual bin sample counts
Expand				Toggle to/from additional information for each range in each bin
Misses				Toggle to/from accumulating Miss Bin count in total sample count and percentages
Write				Write performance analysis results to a disk file (in current display mode)
Help				Get help information for the Performance Analyzer display menu
Run				Run Performance Analysis
Reset (emulator)				Begin performance analysis from RESET conditions (RESET supplied by emulator)
Reset (target)				Begin performance analysis from RESET conditions (RESET supplied by target)
Go				Begin (resume) performance analysis at current PC
Quick-setup				Quick Performance Analyzer setups
Bins	0			Set the number of bins to be used for a Quick-Setup
(Setup Types)				
N-equal				Create 'N' (14) equally sized bins
Module				Create bins using module addresses
Procedure				Create bins using procedure/function addresses
Line numbers				Create bins using source code line number addresses
Misc				Miscellaneous setup options
Clear				Clear performance analyzer setup
Sort Order	RANGE			Toggle the displayed sort order
Capture Span				Specify the code memory capture span
Edit				Edit current bin
Add				Add a bin to Performance Analyzer setup
Delete				Delete the current bin from Performance Analyzer setup
File				Save or Load a Performance Analyzer setup
Save				Save Performance Analyzer setup to a file
Load				Load Performance Analyzer setup from a file
High Bin Count		Alt-F10	HBin PA	High maximum bin count performance analysis (program profiling)
Statistics				Performance Analyzer statistics options
Clear				Clear performance analysis statistics
Accumulate	Off			Accumulate performance analysis statistics toggle
View				View performance analysis statistics (post-emulation)
Raw				Display Mode: bar graph lines only (one per bin)
Symbolic				Display Mode: bar graph lines + all labels (global and/or local) in range
HLL				Display Mode: bar graph lines + HLL source images/lines in ranges in bin
Mixed				Display Mode: bar graph lines + labels + HLL source images (Symbolic + HLL)
Counts				Toggle bar graph lines to/from actual bin sample counts
Expand				Toggle to/from additional information for each range in each bin
Misses				Toggle to/from accumulating Miss Bin count in total sample count and percentages
Write				Write performance analysis results to a disk file (in current display mode)

Help				Get help information for the Performance Analyzer display menu
Run				Run Performance Analysis
Reset (emulator)				Begin performance analysis from RESET conditions (RESET supplied by emulator)
Reset (target)				Begin performance analysis from RESET conditions (RESET supplied by target)
Go				Begin (resume) performance analysis at current PC
Quick-setup				Quick Performance Analyzer setups
Bins	0			Set the number of bins to be used for a Quick-Setup
(Setup Types)				
N-equal				Create 'N' (14) equally sized bins
Module				Create bins using module addresses
Procedure				Create bins using procedure/function addresses
Line numbers				Create bins using source code line number addresses
Misc				Miscellaneous setup options
Clear				Clear performance analyzer setup
Sort Order	RANGE			Toggle the displayed sort order
Capture Span				Specify the code memory capture span
Edit				Edit current bin
Add				Add a bin to Performance Analyzer setup
Delete				Delete the current bin from Performance Analyzer setup
File				Save or Load a Performance Analyzer setup pulldown
Save				Save Performance Analyzer setup to a file
Load				Load Performance Analyzer setup from a file
Source/Symbols				Source level debug and symbol displays
Module Update	AUTO			Modify current module update mode
Auto				Set current module automatically
User				Specify the current module manually
Current Module		Shift-F6	CurMod	Specify the current module manually
Source Path		Shift-F7	SrcPath	View/change the path used for locating HLL source/listing files
Raw Source		Shift-F8	RawSrc	View the raw source for a C module or the listing file for a PL/M-51 module
(Structure)				
Modules		Ctrl-F1	Modules	Display the list of modules in the program
Scopes		Ctrl-F2	Scopes	Display the scopes in the program
Line Numbers		Ctrl-F3	LinNums	Display the line numbers available for (defined in) the program file
(Symbols)				
Global		Shift-F9	SymGlob	Global (PUBLIC) symbol information, sorted alphabetically
Local		Ctrl-F4	SymLocl	Local symbol information, sorted alphabetically
Alpha		Shift-F10	SymAlph	Both global and local symbol information, sorted alphabetically
Address		Ctrl-F5	SymAddr	Global and local symbol information, sorted by address within each memory space
Break/Trace				Modify break-/trace-points; view trace buffer
Set		Shift-F2	SetBkpt	Set simple & complex Break-Points and Trace-Points
Add				Add a break element
CBREAK: Code Break-Point				Add a Code Break-Point element
XBREAK: External Data Break-Point				Add an External Data Break-Point element
TRON: Trace-On-Point				Add a Trace-On-Point element
TROFF: Trace-Off-Point				Add a Trace-Off-Point element
Remove				Remove highlighted break element
Edit				Edit highlighted break element
Hex				Change the display mode to Hexadecimal
Symbolic				Change the display mode to Symbolic
Opcode Class				View/Modify opcode classes
Add				Add an opcode class element
Remove				Remove the current opcode class element
Edit				Edit the current opcode class element
Name				Name opcode class
Mnemonic				Choose opcode mnemonic

Menu item	Shortcut	Description
Operand 1		Choose operand 1 (destination)
None		No operand
Immediate		Select immediate operand
Direct		Select direct operand
Bit		Select bit operand
/Bit		Select complemented bit operand
Code		Select code address operand
A		Select accumulator operand
C		Select carry bit operand
DPTR		Select data pointer operand
PC		Select PC operand
AB		Select AB pair operand
@R0		Select indirect register 0 operand
@R1		Select indirect register 1 operand
Rn		Select register pulldown
R0		Select register 0 operand
R1		Select register 1 operand
R2		Select register 2 operand
R3		Select register 3 operand
R4		Select register 4 operand
R5		Select register 5 operand
R6		Select register 6 operand
R7		Select register 7 operand
R0 - R7		Select register 0 through 7 operand
Operand 2		Choose operand 2 (source)
None		No operand
Immediate		Select immediate operand
Direct		Select direct operand
Bit		Select bit operand
/Bit		Select complemented bit operand
Code		Select code address operand
A		Select accumulator operand
C		Select carry bit operand
DPTR		Select data pointer operand
PC		Select PC operand
AB		Select AB pair operand
@R0		Select indirect register 0 operand
@R1		Select indirect register 1 operand
Rn		Select register pulldown
R0		Select register 0 operand
R1		Select register 1 operand
R2		Select register 2 operand
R3		Select register 3 operand
R4		Select register 4 operand
R5		Select register 5 operand
R6		Select register 6 operand
R7		Select register 7 operand
R0 - R7		Select register 0 through 7 operand
Load		Load opcode class from file
Save		Save opcode class to file
Load		Load break elements from a disk file
Save		Save break elements to a disk file
Clear	ClrBrks	Clear (permanently remove) all simple & complex Break-Points and Trace-Points
Disable	DisBrks	Temporarily remove all simple & complex Break-Points and Trace-Points
Enable	EnaBrks	Restore Break-/Trace-Points temporarily removed by the 'Disable' command above

Trace Trigger	End			Specify when emulation stops, relative to actual break-point, in terms of trace
Start				Emulation stops after 4K trace frames accumulate following break-point
Center				Emulation stops after 2K trace frames accumulate following break-point
End				Emulation stops as soon as the first break-point is reached
Variable	+0			Emulation stops after 'n' trace frames accumulate following break-point
Break-Count	0		BrkCnt	Specify number of break-points to pass through before actually breaking
View Trace		Shift-F3	ViewTrc	View the contents of the trace buffer
Raw				Raw display mode: show trace frame content as address, data and probe
Code				Code display mode: show trace frame content as disassembled instructions
Mixed				Mixed display mode: disassembled instructions with HLL source interspersed
HLL				HLL display mode: show trace frame content as HLL (C or PL/M-51) source lines
Probes				Toggle 'Probes' column between binary, hex & digital waveform display modes
Search				Search trace buffer for a specific code memory address or trace frame number
Write				Write trace buffer data to a file (using current display mode)
Help				Information on various topics

(1): Default value (if applicable)
(2): Default Hot Key (Function Key) assignment
(3): Function Key bottom-of-screen label ("..." if command is not assignable to a Function Key)

Index

!

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

iceMASTER-68HC11 Supplement

The complete *iceMASTER-68HC11* Manual is undergoing final editing and will be available very soon. Until then, you can use the information provided here to augment the information in the *iceMASTER-8051* manual and the online **Help** system.

Description of Files

The following files are supplied on the distribution diskettes for *iceMASTER-68HC11*:

ICE.EXE — The *iceMASTER-68HC11* Host Software program.

$CLR1$ — File containing default video display color and highlighting values suitable for color CGA/EGA/VGA monitors.

$CLR2$ — File containing default video display color and highlighting values suitable for color EGA/VGA monitors.

$CLRM$ — File containing default video display "color" and highlighting values suitable for monochrome monitors.

MF_GEN.EXE — Utility program used to generate a $MODEL file to allow use of a particular emulator probe card. The emulator host software (ICE.EXE) reads the $MODEL file during its initialization sequence.

> The $MODEL file defines all the specific properties of the chip in the probe card (e.g., the 68HC11A8). Such items include register definitions, memory sizes, etc. The $MODEL file also contains the text for all the online Help Topics.

During installation of the Probe Card Distribution Diskette, MF_GEN will be invoked automatically to create the $MODEL file for the probe card you purchased. MF_GEN may be reinvoked manually, if ever necessary, to regenerate the $MODEL file or to generate a different $MODEL file (for a different probe card).

MODELS.DAT — Data file read by the MF_GEN utility program. This file contains the information necessary to generate a $MODEL file for each available probe card.

MFG.EXE — This program is called only from MF_GEN. MFG should never be invoked directly at the DOS command line prompt.

Utility Program Files

HC11BOOT.AOM — This program allows you to emulate the 68HC11's Special Bootstrap operating mode with the *iceMASTER-68HC11* emulator. Detailed instructions for using HC11BOOT.AOM can be found in the assembly language source file (HC11BOOT.S07) or the assembler-generated listing file (HC11BOOT.LST; see page 9 for further details). All the files associated with the HC11BOOT program are:

HC11BOOT.AOM	Loadable, Intel absolute object module format, with symbolic information
HC11BOOT.BAT	Batch file to generate HC11BOOT from scratch
HC11BOOT.HEX	Loadable, standard hex format

HC11BOOT.LST	Assembler output listing
HC11BOOT.MAP	Linker memory map
HC11BOOT.S07	Assembly language source
HC11BOOT.S19	Loadable, Motorola S-Record format

Simple Assembly Language Demonstration Program

DEMO_H11.AOM	Loadable, Intel absolute object module format, with symbolic information
DEMO_H11.BAT	Batch file to generate DEMO_H11 from scratch
DEMO_H11.HEX	Loadable, standard hex format
DEMO_H11.LST	Assembler output listing
DEMO_H11.MAP	Linker memory map
DEMO_H11.S07	Assembly language source
DEMO_H11.S19	Loadable, Motorola S-Record format

Simple IAR/Archimedes C Language Demonstration Program

H_CSTART.LST	Assembler-generated listing file for H_CSTART.S07
H_CSTART.S07	Assembly Language source module (C runtime library startup module)
H_DEMO.AOM	Loadable, absolute object module format, (with debug information)
H_DEMO.H	C language *#include* file
H_DEMO.MAP	Linker-generated map of H_DEMO.AOM
H_HLMAIN.C	Main program (function 'main()'; C source)
H_HLMAIN.LST	Compiler-generated listing file for H_HLMAIN.C
H_INNER.C	Function 'innerloop()' (C source)
H_INNER.LST	Compiler-generated listing file for H_INNER.C
H_MAKE.BAT	Batch file to generate H_DEMO.AOM from scratch
H_MAKE.LCL	Librarian control file for generating H_DEMO.AOM
H_MAKE.XCL	Linker control file for generating H_DEMO.AOM
H_WASTE.C	Function 'wastetime()' (C source)
H_WASTE.LST	Compiler-generated listing file for H_WASTE.C

Register Bit Names

To further enhance the *iceMASTER-68HC11* emulator, we have supplied special symbols in the $MODEL file for the bit names in those registers which have special functions assigned to different bits in the register. These symbols are the standard names preceded by an at-sign (@). These symbols are special because encoded in their internal addresses are both the byte offset of the register into the Register Block and the mask to denote the bit within the register. When browsing or perusing the Register Window (*Configure* | *Windows* | *Goto* command), a box will pop-up over, for example, SCCR2,

```
─────────────────────────────────Registers──────────────────────────────68HC11A0─
  A:00  B:00  X:0000  Y:0000  SP:0094  CCR:00  S:0 X:0 H:0 I:0 N:0 Z:0 V:0 C:0
 ↑PIOC--07    PORTA--8F    PORTB--00    PORTC--FF    PORTCL-00    PORTD--7F
 ↓PORTE--FF   PPROG--00    SCCR1--C0    SCCR2--00    SCDR---FF    SCSR---C0
─────────────────────────────────Sta ┌─────102D─────────┐──────────SP:0094─
 ↕0094 FF FE F6 FF FF FF FF FF FF FF FF │80 SCCR2.7 @TIE   0│ FF FF FF FF FF FF
 ─Module:H_HLMAIN─────────────────Sou │40 SCCR2.6 @TCIE  0│amples\h_demo.aom─
 FEF0 8E0096     STARTUP:    LDS   #$0 │20 SCCR2.5 @RIE   0│     SP=0094 n0z0v0
 FEF3 BDFEF9                 JSR   MAI │10 SCCR2.4 @ILIE  0│    ↓FEF9 SP=0096
 FEF6 7EFFB7                 JMP   ?C_ │08 SCCR2.3 @TE    0│
 H_HLMAIN:#28      state = 0;          │04 SCCR2.2 @RE    0│
 FEF9 5F         MAIN:       CLRB      │02 SCCR2.1 @RWU   0│    B=00 n0z0v0c0
 FEFA F7FFBA                 STAB  STA │01 SCCR2.0 @SBK   0│
 H_HLMAIN:#29      DDRD |= DDD0;       └──────────────────┘ection for PORTD.0
 FEFD C601                   LDAB  #$01                     ;H_HLMAIN:#29
 FEFF 37                     PSHB
 FF00 CC1009                 LDD   #$1009
─────────────────────────────────Status───────────────────────────────────
  BrkCnt:0     Time:                  0µs State:Break-point    SP:0094 PC:FEF0
  RepCnt:1     Resets:0         Trace:Partial Read:100% Trig:End    BrkAddr:F500
──────────────────────────────────────────────────────────────────────────
Tab/Shift-Tab:Next/Prev window  (Keypad):Scroll  (Number)Enter:Change  Esc:Exit
sLoad    SetBkpt ViewTrc DisAssm MacExec CurMod    SrcPath RawSrc   SymGlob1SymAlph
1Help   2ResetEm3ResetPr4Go     5GoFrom 6GoUntil7StepIns8StepLin9StepOvr0StepTo
```

Figure 1

showing all the bits defined in that register (e.g., @RE). To toggle a bit's value, all you have to do is type the name of the bit. We preceded these names with the at-sign (@) so that these predefined symbols would not conflict with any symbolic information in your program.

iceMASTER-68HC11 always displays the individual bits of the CCR (Condition Code Register) in the top line of the Register Window. The individual bits in the CCR can be referenced symbolically using their standard names (e.g., N).

INIT Register Configuration

The 68HC11's INIT Register specifies the starting addresses of the on-chip RAM and Register Blocks. By default, the RAM block begins at 0 and the Register Block begins at 4K ($1000). If your application program requires different starting addresses for either the RAM or the Register Blocks, you can use the *Configure|Emulator|INIT Register* command to pop up a menu allowing you to set up the emulator to use a different starting location for either one, or both, of these on-chip blocks. In addition, your

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
                                 Registers                       68HC11A0
 Emulator             0  Y:0000  SP:0000  CCR:D0  S:1 X:1 H:0 I:1 N:0 Z:0 V:0 C:0
                         ADR2--00     ADR3---00     ADR4---00     BAUD---07
  Execute                COPRST-00    DDRD---00     HPRIO--25     INIT---01
                                   Stack                               SP:0000
  Comm port      COM1 ▶ F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
  Baud rate    115200 ▶          Source                   File:demo_h11.aom
  INIT register    01 ▶     LDX   #$1000                       X=0000 n0z0v0
  EEPROM             68HC11 INIT Register Configuration
                                                       INIT Register = $01
 Function-K  RAM Block Starts at $0000 ($0100/256 bytes):
 Save          0K  4K  8K 12K 16K 20K 24K 28K 32K 36K 40K 44K 48K 52K 56K 60K
 Restore
             Register Block Starts at $1000 ($0040/64 bytes):
 FF94 8D19     0K  4K  8K 12K 16K 20K 24K 28K 32K 36K 40K 44K 48K 52K 56K 60K
 FF96 8601   Enter:Accept                                          Esc:Cancel
 FF98 A808                  EORA  $08,X
                                 Status
 BrkCnt:0         Time:                3µs  State:Break-point   SP:0000 PC:FF83
 RepCnt:1         Resets:0     Trace:Partial Read:100% Trig:End     BrkAddr:FF83

 ←→:Set start address for RAM Block      ↑↓:Modify Register Block start address
sLoad     SetBkpt  ViewTrc  DisAssm  MacExec  CurMod   SrcPath  RawSrc   SymGlob 1SymAlph
1Help    2ResetEm 3ResetPr 4Go      5GoFrom  6GoUntil 7StepIns 8StepLin 9StepOvr 0StepTo
```

Figure 2

application program is responsible for setting the INIT register to the same value (if you start emulation using any of the emulation reset commands: *Run|Reset|Normal|Emulator*, *Run|Reset|Normal|Target*, *Run|Reset|Special|Emulator* or *Run|Reset|Special|Target*).

Each time emulation breaks, the host software and emulator firmware perform several validity checks to ensure that the INIT register does indeed contain the value specified here.

To actually effect the change to the INIT register, you must select the *Configure|Emulator|Execute* command. As part of the sequence of steps initiated by the *Configure|Emulator|Execute* command, the following code fragment is executed, starting from a Normal Mode reset, in the 68HC11 processor in the probe card:

```
86xx      LDAA  #init_value     value specified in menu
B7103D    STAA  $103D           default location of INIT
```

This is done primarily so that any Windows into the on-chip RAM (e.g., the Stack Window) or the Register Block (e.g., the Register Window) are painted initially with the current, correct values before you actually load a program and begin emulation.

The *Configure|Emulator|INIT Register* pop-up menu also shows the location and size of any other on-chip memory resource which may be, or have become, enabled (visible). This information is displayed at the bottom of the menu, as shown in Figure 3. Note that the menu in Figure 3 is shown for

```
Configure  File  Run  Display/Alter  Misc  Source/Symbols  Break/Trace  Help
  Emulator        ▶ 0  Y:0000  SP:0096  CCR:00  S:0 X:0 H:0 I:0 N:0 Z:0 V:0 C:0
                        CONFIG·0F    COPRST·00    DDRC···00    DDRD···00
   Execute              OC1D···00    OC1M···00    OPTION·10    PACNT··00
                   ----------------Stack-------------------------SP:0096
   Comm port      COM1 ▶ F FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
   Baud rate    115200 ▶ ------Source--------File:c:\examples\h_demo.aom
   INIT register    01 ▶     LDS   #$0096                   SP=0000 n0z0v0
              ==========68HC11 INIT Register Configuration==========
  Options                                             INIT Register = $01
  Function-K  RAM Block Starts at $0000 ($0200/512 bytes):
  Save           0K   4K  8K 12K 16K 20K 24K 28K 32K 36K 40K 44K 48K 52K 56K 60K
  Restore
              Register Block Starts at $1000 ($0040/64 bytes):
 FEFD C601       0K   4K  8K 12K 16K 20K 24K 28K 32K 36K 40K 44K 48K 52K 56K 60K
 FEFF 37
 FF00 CC1009  CONFIG:  EEPROM          Block Starts at $B600 ($0200/512 bytes)
              HPRIO:   Bootstrap ROM Block Starts at $BF40 ($00C0/192 bytes)
  BrkCnt:0    CONFIG:  Code ROM        Block Starts at $D000 ($3000/12288 bytes)
  RepCnt:1    Enter:Accept                                        Esc:Cancel

  ↔:Set start address for RAM Block      ↑↓:Modify Register Block start address
sLoad     SetBkpt  ViewTrc  DisAssm  MacExec  CurMod   SrcPath  RawSrc   SymGlob 1SymAlph
1Help    2ResetEm 3ResetPr 4Go      5GoFrom  6GoUntil 7StepIns 8StepLin 9StepOvr 0StepTo
```

Figure 3

illustrative purposes only. You should not operate the *iceMASTER-68HC11* with either the on-chip Bootstrap ROM enabled or the on-chip ROM enabled.

iceMASTER-68HC11 Emulation Environment Requirements

HPRIO.MDA = 1

Normal Expanded or Special Test mode in effect. The 68HC11 processor in the probe card must physically be operated in either the Normal Expanded mode or the Special Test mode. The *iceMASTER-68HC11* emulator supports the Normal modes (Expanded or Single-Chip) via two different probe cards: a probe card for Expanded mode and a probe card for Single-Chip mode. Either of these probe card types (Expanded or Single-Chip) can also be operated in the Special Test mode. The Special Bootstrap mode is supported via the supplied utility program HC11BOOT.AOM; commentary at the beginning of HC11BOOT.S07 (source) and HC11BOOT.LST (listing) describes how to use the *iceMASTER-68HC11* emulator to execute in the Special Bootstrap mode environment (see page 9 for further details).

CONFIG.ROMON = 0

On-chip program ROM disabled. You cannot execute in the on-chip ROM. If you should inadvertently set the ROMON bit in the CONFIG register, the host software will detect this situation when you select the *Configure|Emulator|Execute* command and attempt to recover. The recovery action consists of rewriting the CONFIG cell/register, with the ROMON bit turned off, and then resetting the 68HC11 part. This is the only time that the emulator automatically writes to the CONFIG cell. If you desire to make any other changes to the CONFIG cell/register, you can make them directly using either the *Display/Alter|Var/Reg* command, or the *Configure|Windows|Goto* command and position into the Register Window – see page 8.

HPRIO.RBOOT = 0

On-chip Bootstrap ROM disabled. You cannot execute in the on-chip Bootstrap ROM. However, see the utility program HC11BOOT (page 9), which copies the Bootstrap ROM code from the on-chip Bootstrap ROM into emulation code memory, whence the Bootstrap code can then be executed in Special Test mode.

HPRIO.IRV = 0

Internal Read Visibility disabled. We recommend that you run with IRV disabled.

Miscellaneous

Mapping

The host software will force any enabled (visible) on-chip memory resource (e.g., on-chip RAM, Register Block, on-chip EEPROM) to be mapped to the emulator. Such areas cannot be mapped to memory in your target system. After each emulation cycle, the emulator firmware and host software check to see if the status of any on-chip memory resource has changed (enabled vs. disabled). If so, the emulator will adjust the mapping automatically to account for the change.

Note: Even though the on-chip memory area is reported as being mapped to the emulator base, the memory area actually referenced during emulation is the on-chip memory resource itself.

Break-Points

If a **CBREAK: Code Break-Point** is set on an instruction, emulation breaks (stops) <u>before</u> that instruction executes. The same holds true for Trace-On-Points and Trace-Off-Points points.

A **WBREAK: Write Access Break-Point** at a particular location causes emulation to break (stop) <u>after</u> execution of an instruction writing to (storing into) that location. In such a case, the memory location <u>is</u> updated (i.e., the write is <u>not</u> suppressed). In addition, if the program's execution flow happens to pass through that location, emulation breaks <u>before</u> fetching the opcode from that location. Write Access Break-Points are available only in the Simple Break window.

A **PBREAK: Write Protect Break-Point** on a particular memory location causes emulation to break <u>after</u> execution of an instruction which attempts to write to (store into) that memory location. In such a case, the memory location is <u>not</u> updated (i.e., the write is suppressed). Write Protect Break-Points can only be set for a memory location/range mapped to the emulator base, not for a memory location/range mapped to the target system. Additionally, Write Protect Break-Points are available only in the Simple Break window.

You can execute code in either the on-chip RAM or the on-chip EEPROM. However, you cannot set Code or Write Protect break-points (using the *Break/Trace|Set* menu) in any of the enabled on-chip memory areas. These include the on-chip RAM, on-chip Register Block and on-chip EEPROM. If the processor is executing in an on-chip memory area (RAM or EEPROM) when a break in emulation occurs (e.g., by pressing the **Break** button on the emulator base or by pressing the **Esc** key), the software will warn you that such a condition has occurred and that the 68HC11 processor in the probe card has been <u>reset</u>. In such cases, you can use the Trace Buffer to find out where the program was executing when emulation stopped.

The following table summarizes which type of break-/trace-points can be used in the various memory areas:

Break-/Trace-Point Type	Break-/Trace-Point Availability				
	On-Chip Memories			64K Emulation Memory	64K Target Memory
	RAM	Registers	EEPROM		
CBREAK Code Break-Point	No	No	No	Yes	Yes
PBREAK Write Protect Break-Point	No	No	No	Yes	No
WBREAK Write Access Break-Point	Yes	Yes	Yes	Yes	Yes
TRON Trace-On-Point	Yes	Yes	Yes	Yes	Yes
TROFF Trace-Off-Point	Yes	Yes	Yes	Yes	Yes

Stack

If the Stack Pointer (SP Register) is pointing into the on-chip RAM before emulation begins, the next unused byte at the top of the stack will be changed (written) upon restarting emulation. This is a side effect of needing to restore the A Register without affecting any of the bits in the Condition Code Register (CCR).

If the SP register is pointing into any other memory area (e.g., off-chip RAM), no memory contents will be changed when emulation is restarted.

File Loading

You use the *File | Load* command to load a file into the emulator, initializing memory with the image of your program. This can be either the emulator's memory or memory in your target system, or both, depending on the current mapping (*Configure | (Mapping)Code Memory* command).

If the file contains bytes which are destined for the on-chip RAM, the on-chip RAM will be initialized with those bytes.

In addition, if the on-chip EEPROM is enabled, any bytes in the file destined for the on-chip EEPROM will be written there using the standard "erase-before-write" method, one byte at a time. However, due to optimizations employed during file loading, we recommend that, if your program file contains initialization for EEPROM, you first use the *Display/Alter | Code | Fill* command to fill all of EEPROM with $FF bytes before actually loading the file.

On-Chip Memories

When a value is written to an on-chip memory (e.g., on-chip RAM or Register Block), the corresponding location in the 64K memory in the emulator base unit is also written with the same value.

Writing EEPROM

Whenever the *iceMASTER-68HC11* writes to on-chip EEPROM memory, it pops up a box showing which locations in EEPROM are being written. This is done because writes to an EEPROM location take considerably longer than writes to other memories, due to the "erase-before-write" technique used and the need to delay 10 milliseconds after each erase and write operation.

The only time *iceMASTER-68HC11* writes to EEPROM memory is when the on-chip EEPROM is enabled and you explicitly request the write to take place (e.g., by using the *Display/Alter | Code | Fill* command or by loading a file which initializes all or part of the EEPROM).

Writing the CONFIG Cell/Register

You can change the value in the CONFIG cell/register, subject to the normal operating restrictions of the 68HC11 processor in the probe card. That is, you must first put the processor into Special Test Mode in order to be able to write to the CONFIG cell (an EEPROM location in most parts). In addition, if the processor has a BPROT register, you must at least set the PTCON bit in the BPROT register to zero before attempting to update the CONFIG cell. Finally, you will not see any immediate confirmation that the new value was actually written to the CONFIG cell; you must cause the processor to go through a reset in order to have the value in the CONFIG cell copied into the CONFIG register.

Disassemblies

In disassemblies, several opcode mnemonics have aliases (alternative mnemonic names). In such cases, the alternative name will be shown as a comment next to the disassembled instruction:

Mnemonic	Alias
ASLD	LSLD
BCC	BHS
BCS	BLO
ASLA	LSLA
ASLB	LSLB
ASL	LSL

You can use either form shown above when patching in new instructions using the single-line assembler (*Display/Alter|Asm/Dasm|Assemble* command).

Additionally, the *Display/Alter|Asm/Dasm|Disassemble* command disassembles words (byte pairs) in the current Interrupt Vector area as

```
ivect <address>
```

The current Interrupt Vector area is determined by the current operating mode of the 68HC11 processor in the probe card. The Special Modes Interrupt Vectors are located at $BFC0 through $BFFF. The Normal Modes Interrupt Vectors are located at $FFC0 through $FFFF.

Status Window

The top right hand border of the Status Window will show "Mode:Special" whenever the 68HC11 processor in the probe card is in Special Test mode (HPRIO.MDA = 1) at a break-point.

View Trace

In the *Break/Trace|View Trace* menu, when the display mode is either Code or Mixed, a small box may pop up at the currently highlighted instruction in the Trace Buffer. This box contains information extracted from the trace frames for the execution of that instruction. This information consists of values of locations/registers manipulated by that instruction. The top border of the box may show "(updated value)" to indicate that the value shown in the box is the value after the instruction executed:

View Trace Pop-Up Box Content	Applicable Instructions
`mem[xxxx]=xx`	{many}
`mem[xxxx]=xxxx`	{many}
`mem[xxxx]=xx; result=xx`	{many}
`SP=xxxx`	TSX,TSY
`CCR=xx,B=xx,A=xx,X=xxxx,Y=xxxx,Ret=xxxx`	RTI,SWI

TEST Instruction

If the TEST instruction (opcode value $00) is executed during emulation in Special Test Mode, a box will pop up informing you that the processor in the probe card is inactive. If you press Esc, the host software will ask you whether or not you want to reset the 68HC11 processor to break (stop) emulation.

ACTIVE Light

When the green **ACTIVE** light on the emulator base is on, it indicates that the LIR and AS signals in the 68HC11 are toggling properly.

XIRQ

When emulation stops, XIRQ is disabled. When emulation begins/resumes, XIRQ is enabled.

Reading Ports

The origin of Port values displayed in the Register Window depends on the configuration of each port pin. If the pin is configured as an output pin, the value displayed is that from the port register. If the pin is configured as an input pin, the value displayed is that from the port pin.

Special Bootstrap Mode

The HC11BOOT.AOM utility program allows you to use the *iceMASTER-68HC11* emulator to execute in the Special Bootstrap mode environment. The listing for HC11BOOT appears on the following pages. The commentary box at the beginning of the listing provides detailed instructions for using HC11BOOT.

```
################################################################################
#                                                                              #
#    Archimedes 6801 Assembler V1.85/MD2               09/Aug/91   04:07:55   #
#                                                                              #
#    Source   =  hc11boot.s07                                                 #
#     List    =  hc11boot.lst                                                 #
#     Object  =  hc11boot.r07                                                 #
#    Options  =  s x                                                          #
#                                                                              #
#                                 (c) Copyright Archimedes Software Inc. 1987 #
################################################################################

                    * Source File:  HC11BOOT.S07

       0000                 NAME    HC11BOOT
       0000                 TITL    'HC11BOOT -- Prepare for Bootstrap Loading'
       0000                 P68H11  Enable the HC11 instruction set

                    ******************************************************************************
                    * The HC11BOOT program allows you to emulate the 68HC11's
                    * Special Bootstrap operating mode with the iceMASTER-68HC11:
                    *
                    *    1.  Select the 'Run|Reset|Special|Processor' command.
                    *           This resets the 68HC11 processor in the probe card into
                    *           Special Test mode.  The only reason for doing this is so
                    *           that, after loading the HC11BOOT program in step 2 below,
                    *           the host software can "tell" you where execution will begin.
                    *           Because the processor is in a Special mode rather than a
                    *           Normal mode, execution from a reset will begin at the address
                    *           contained in the Special Modes Reset Vector ($BFFE), rather than
                    *           at the address contained in the Normal Modes Reset Vector ($FFFE).
                    *           This is only an issue with respect to painting the Source and
                    *           Status Windows to reflect the (assumed) current PC value after
                    *           a new program has been loaded into the emulator.
                    *    2.  Select the 'File|Load' command:
                    *           a.  File to load:               HC11BOOT.AOM (this program)
                    *           b.  Merge into current
                    *               application environment?:  No
                    *    3.  Select the 'Break/Trace|Set|Add|CBREAK: Code Break-Point' command:
                    *           a.  Set a single, simple break-point at:  BRKADDR
                    *    4.  Select the 'Run|Reset|Special|Emulator' command.
                    *           This begins execution of this program (HC11BOOT) from a reset
                    *           condition, running in Special Test mode.  This program copies
                    *           the bootstrap loader from the on-chip Bootstrap ROM into the
                    *           corresponding memory locations in the emulator's code memory.
                    *           After copying the bootstrap loader, this program reaches the
                    *           break-point at 'brkaddr'.
                    *    5.  Select the 'Break/Trace|Clear' command to remove all break-points
                    *        (specifically, the simple break-point at BRKADDR).
                    *    6.  Select the 'Run|Reset|Special|Emulator'
                    *        or     the 'Run|Reset|Special|Target' command
                    *        depending on your target environment configuration.
                    *           This begins execution of the 68HC11 bootstrap loader from a
                    *           reset condition, running in Special Test mode.
                    *
                    * NOTE that HC11BOOT modifies the Special Modes Reset Vector during
                    * execution.  This happens because it copies the Reset Vector from the
                    * Bootstrap ROM into the Special Modes Reset Vector in emulation code
                    * memory.  Thus, if you need to run HC11BOOT several times in succession,
                    * you must 'File|Load' the HC11BOOT program each time (i.e., repeat steps
                    * 1 through 6 above each time).
                    ******************************************************************************


                    * Register Equates

       1000         REGBAS  EQU     $1000    starting address for Register Block
       003C         HPRIO   EQU     $3C      Highest Priority Interrupt and Miscellaneous Register:
       0080         RBOOT   EQU     $80         HPRIO.7 On-chip Bootstrap ROM enabled
                    * Misc
       00B0         STKBAS  EQU     $00B0    beginning of stack in on-chip RAM
       BF40         brom_ba EQU     $BF40    Bootstrap ROM beginning address (chip-dependent):
                    *                           A0,A1,A2,A8,E0,E1,E2,E9:  $BF40
                    *                           D3,F1:                    $BF00
       BFFF         brom_ea EQU     $BFFF    Bootstrap ROM ending     address (never changes)
       00C0         brom_sz EQU     (brom_ea-brom_ba)+1  # of bytes in Bootstrap ROM
       C000         buf_ba  EQU     brom_ea+1         beginning addr of temp buffer for boot code
       00C0         buf_sz  EQU     brom_sz           # of bytes in temp buffer for boot code
                    *                                    (same size as Bootstrap ROM)
       C0BF         buf_ea  EQU     buf_ba+(buf_sz-1) ending    addr of temp buffer for boot code
```

```
COCO            porg    EQU     buf_ea+1          program origin

                * Memory Layout for HC11BOOT:
                *               _____
                * $0000          |-RAM Block       {default location --
                * $(RAMend)______|                  this program uses none}
                * $(RAMend+1)    |-(not used by this program)
                * $0FFF__________|
                * $1000          |-Register Block  {default location --
                * $(Regend)______|                  this program uses only HPRIO}
                * $(Regend+1)    |-(not used by this program)
                * $2000          |
                * $3000          |
                * $4000          |
                * $5000          |
                * $6000          |
                * $7000          |
                * $8000          |                         Step 1:     Step 2:
                * $9000          |                         Bootstrap   Bootstrap
                * $A000          |                         ROM On      ROM Off
                * $BF3F/BEFF_____|                          ____         ____
                * $BF40/BF00     |-Bootloader Code Bytes   |    |       |    |
                *                |   (in Bootstrap ROM     |from|---+ +-->| to |
                * $BFFF__________|    and emulator's RAM)  |____|   | |   |____|
                * $C000          |-Buffer into which Bootstrap      | |
                *                |   ROM bytes are first copied | to |<--+ +---|from|
                * $C0BF/C0FF_____|   from on-chip Bootstrap ROM |____|      |____|
                * $C0C0/C100     |-This program (HC11BOOT)
                * $C0F9/C139_____|
                * $C0FA/C13A     |-(not used by this program)
                * $D000          |
                * $E000          |
                * $F000          |
                * $FFFF__________|

COCO                    ORG     porg
COCO            start:
COCO 8E00B0             LDS     #STKBAS          establish top of stack

                * 1.  Copy on-chip Bootstrap ROM bytes, including Special Modes
                *     Interrupt Vectors, to temporary buffer in emulator's code memory:

C0C3 CE1000             LDX     #REGBAS
C0C6 1C3C80             BSET    HPRIO,X,#RBOOT  enable on-chip Bootstrap ROM (make it visible)
C0C9 CEBF40             LDX     #brom_ba        starting address:  from
C0CC 18CEC000           LDY     #buf_ba         starting address:  to

C0D0 A600       cpy1lp: LDAA    0,X             loop
C0D2 18A700             STAA    0,Y              to
C0D5 08                 INX                      copy
C0D6 1808               INY                      one
C0D8 8CC000             CPX     #brom_ea+1       byte
C0DB 25F3               BLO     cpy1lp           at a time

                * 2.  Copy Bootstrap ROM bytes and Interrupt Vectors from temporary buffer
                *     in emulator's code memory to the "normal" locations for the boostrap
                *     loader code (but copy into the emulator's code memory, not into the
                *     on-chip Bootstrap ROM):

C0DD CE1000             LDX     #REGBAS
C0E0 1D3C80             BCLR    HPRIO,X,#RBOOT  disable on-chip Bootstrap ROM (make invisible)
C0E3 CEC000             LDX     #buf_ba         starting address:  from
C0E6 18CEBF40           LDY     #brom_ba        starting address:  to

C0EA A600       cpy2lp: LDAA    0,X             loop
C0EC 18A700             STAA    0,Y              to
C0EF 08                 INX                      copy
C0F0 1808               INY                      one
C0F2 188CC000           CPY     #brom_ea+1       byte
C0F6 25F2               BLO     cpy2lp           at a time

                * With a break-point set at 'brkaddr', when emulation breaks,
                * you should select the 'Run|Reset|Special|Emulator' or
                *                       'Run|Reset|Special|Target' command to begin
                * execution of the bootstrap loader.  We do this, rather than jumping
                * directly to 'brom_ba' here, in case there is any assumption/requirement
                * within the bootloader code that it begin execution from "pure" reset
                * conditions.

C0F8            brkaddr:
```

```
148  C0F8 20FE          BRA    *
149
150                 * Establish the actual Reset Vector:
151  BFFE               ORG    $BFFE       Special Modes Reset Vector address
152  BFFE C0C0          FDB    start
153  BFFA               ORG    $BFFA       Special Modes COP Failure
154  BFFA C0C0          FDB    start
155  BFFC               ORG    $BFFC       Special Modes Clock Monitor Fail
156  BFFC C0C0          FDB    start
157
158  BFFE               END

    Errors:  None       ############
    Bytes:   64         # HC11BOOT #
    CRC:     BB92       ############

                 Symbol and Cross Reference Table
                 ================================

Symbol      Value Type  Defline   Refline

                        Segment Definitions
                        ===================


                        External Symbols
                        ================


                        Public Symbols
                        ==============


                        Local Symbols
                        =============

HPRIO       003C  A       55       111   128

RBOOT       0080  A       56       111   128

REGBAS      1000  A       54       110   127

STKBAS      00B0  A       58       105

brkaddr     C0F8  A      147

brom_ba     BF40  A       59        63   112   130

brom_ea     BFFF  A       62        63    64   119   136

brom_sz     00C0  A       63        65

buf_ba      C000  A       64        67   113   129

buf_ea      C0BF  A       67        68

buf_sz      00C0  A       65        67

cpy1lp      C0D0  A      115       120

cpy2lp      C0EA  A      132       137

porg        C0C0  A       68       103

start       C0C0  A      104       152   154   156

                        Macro Definitions
                        =================
```

iceMASTER-68HC11 Probe Cards

Processor Used

The processor actually used in each probe card is as follows:

Probe Card Type	Processors Supported	Processor Used in Probe Card
68HC11 Expanded Mode	68HC11A0 68HC11A1	68HC11A1
	68HC11D0	68HC11D0
	68HC11E0 68HC11E1	68HC11E1
	68HC811E2	68HC811E2
68HC11 Single-Chip Mode	68HC11F1	68HC11F1
	68HC11A7 68HC11A8	68HC11A1
	68HC11D3 68HC711D3	68HC11D0
	68HC11E8 68HC11E9 68HC711E9	68HC11E1
	68HC811E2	68HC811E2

All probe cards are shipped from the factory with the on-chip EEPROM memory disabled (CONFIG.EEON = 0) and with the COP (Computer Operating Properly) WDT (Watchdog Timer) disabled (CONFIG.NOCOP = 1). Additionally, for those parts having a moveable EEPROM (e.g., 68HC811E2 and 68HC11F1), the CONFIG register is set up so that if the EEPROM is enabled, the EEPROM initially will not be at the high end of memory (i.e., not covering the Normal Modes interrupt/reset vectors).

Jumper Blocks

All *iceMASTER-68HC11* probe cards, whether Expanded mode or Single-Chip mode, have only one user-selectable jumper block:

XTAL – Clock Source

The function of the **XTAL Jumper** is to select the source of the clock (oscillator) for the 68HC11:

PC: Probe card's crystal is used.

TAR: Target system supplies crystal or external clock.

This is a double jumper to ensure correct configuration. The center post is the common post.

68HC11Ex Probe Cards

We use the 68HC11E1 chip in the probe cards which support the following devices:

68HC11E0

68HC11E1

68HC11E8

68HC11E9

68HC711E9

The 68HC11E1 physically has an EEPROM/CONFIG Block Protect Register – BPROT. BPROT is initialized out of Reset (every time) to prevent writing to the EEPROM memory and to the CONFIG cell. The 68HC11E0 and 68HC11E8 chips do not have a BPROT Register (i.e., they do not have this secondary level of protection against writing to some EEPROM cell).

If you are developing a 68HC11E0 or 68HC11E8 application using the *iceMASTER-68HC11* emulator, you must account for the fact that the BPROT Register physically exists in the chip in the probe card. The simplest solution is to add code which clears (zeroes-out) the BPROT Register immediately after a Reset. If you wish, you can even leave this BPROT-clearing code in when you go into production with your application. Writing to a non-existent register in the Register Block is harmless in most cases.

68HC11Dx Probe Cards

When using a probe card supporting the 68HC11D0, 68HC11D3 or 68HC711D3, you must explicitly clear the ROMON bit in the CONFIG register after every reset. The default state of the ROMON bit in the CONFIG register following a reset is 1 (ROM enabled).

Single-Chip Mode Probe Cards

All *iceMASTER-68HC11* Single-Chip mode probe cards use either the MC68HC24 or MC68HC27 Port Replacement Unit to recreate the single-chip mode functions of Ports B and C. The resultant timing characteristics of the probe card reflect the internal timing in the MC68HC24/MC68HC27 device with respect to STRA, STRB, PORTB and PORTC. Refer to the appropriate data book (MC68HC24 or MC68HC27) for complete details.

The *iceMASTER-68HC11* host software forces the registers at \$x002-\$x007 to appear to be mapped to your target system. This is done only to support the operational characteristics of the MC68HC24 or MC68HC27 Port Replacement Unit in the probe card.

Finally, for all Single-Chip mode probe cards, it is safest to initialize the INIT register <u>explicitly</u> after every reset. The reason is that even though the INIT register in the 68HC11 part is time-protected in Normal Expanded mode, the mirror INIT register maintained in the MC68HC24/MC68HC27 PRU part is a <u>write-once</u> register. That is, this mirror INIT register can be written at any time after each reset, but it can only be written once after each reset. Normally, this would present no problem. However, if the first attempt to write to the INIT register occurs later than the first 64 cycles following a reset, the new value will be written to the mirror INIT in the PRU but not to the INIT in the 68HC11. Conversely, you should write to the INIT register <u>only</u> once following a reset. Again, the mirror INIT in the PRU can be written only once following a reset; all other writes to this mirror INIT, even if they occur within 64 cycles of the reset, will be ignored (unlike the 68HC11 part itself).

iceMASTER-COP8 Supplement

The complete *iceMASTER-COP8* Manual is undergoing final editing and will be available soon. Until then, you can use the information provided here to augment the information in the *iceMASTER-8051* manual and the online **Help** system.

Description of Files

The following files are supplied on the distribution diskettes for *iceMASTER-COP8*:

ICE.EXE — The *iceMASTER-COP8* Host Software program.

$CLR1$ — File containing default video display color and highlighting values suitable for color CGA/EGA/VGA monitors.

$CLR2$ — File containing default video display color and highlighting values suitable for color EGA/VGA monitors.

$CLRM$ — File containing default video display "color" and highlighting values suitable for monochrome monitors.

MF_GEN.EXE — Utility program used to generate a $MODEL file to allow use of a particular emulator probe card. The emulator host software (ICE.EXE) reads the $MODEL file during its initialization sequence.

> The $MODEL file defines all the specific properties of the chip in the probe card (e.g., the COP880). Such items include register definitions, memory sizes, etc. The $MODEL file also contains the text for all the online Help Topics.

During installation of the Probe Card Distribution Diskette, MF_GEN will be invoked automatically to create the $MODEL file for the probe card you purchased. MF_GEN may be reinvoked manually, if ever necessary, to regenerate the $MODEL file or to generate a different $MODEL file (for a different probe card).

MODELS.DAT — Data file read by the MF_GEN utility program. This file contains the information necessary to generate a $MODEL file for each available probe card.

MFG.EXE — This program is called only from MF_GEN. MFG should never be invoked directly at the DOS command line prompt.

$CR_C880
$CR_C888 — Emulator Control RAM initialization files. The host software automatically downloads one of these files (depending on which probe card you are using) into the emulator base unit during initialization.

MKSHF Utility Program Files

MKSHF.EXE — Utility program to **M**a**K**e a **S**ymbolic ".**H**EX" **F**ile, using as inputs the symbol (.SYM) file produced by the National's COP800 Assembler and the loadable (.HEX) file produced by National's LMHEX utility. The MKSHF utility is described in more detail on page 4.

MKSHF.DOC — Describes how to use the MKSHF.EXE utility program.

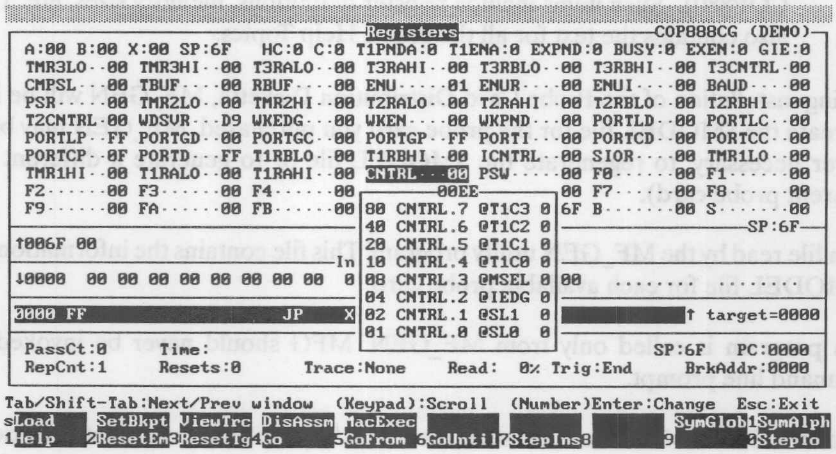
MKSHF.EXC — Standard "exception" file which can be used in creating a customized exception file for input to the MKSHF utility.

Simple Demonstration Program

DEMO_C8.ASM	COP8 demo program (COP800 assembly language source)
DEMO_C8.BAT	Batch file to create loadable DEMO_C8.SHF from scratch
DEMO_C8.EXC	Exception File (input to MKSHF.EXE) for DEMO_C8
DEMO_C8.HEX	LMHEX-generated ".HEX" file for DEMO_C8
DEMO_C8.LM	Assembler-generated load module file for DEMO_C8
DEMO_C8.LST	Assembler-generated listing file for DEMO_C8
DEMO_C8.SHF	Symbolic HEX File for DEMO_C8 (loadable into *iceMASTER*)
DEMO_C8.SYM	Assembler-generated symbol file for DEMO_C8

Special Bit Names for Some Registers

To further enhance the *iceMASTER-COP8* emulator, we have supplied special symbols in the $MODEL file for the bit names in those registers which have special functions assigned to different bits in the register. These symbols are the standard names preceded by an at-sign (@). These symbols are special because encoded in their internal addresses are both the byte address of the register and the mask to denote the bit within the register. When browsing or perusing the Register Window ("Configure|Windows|Goto" command), a box will pop-up over, for example, CNTRL, showing all



the bits defined in that register (e.g., @IEDG). To toggle a bit's value, all you have to do is type the name of the bit. We preceded these names with the at-sign (@) so that these predefined symbols would not conflict with any symbolic information in your program.

iceMASTER always displays the individual bits of the PSW register in the top line of the Register Window. Because there is special processing internally to do this, the individual bits in the PSW register can be referenced symbolically using either their standard names (e.g., HC) or using the @-names (e.g., @C).

Rn (Fn) Registers

Generated $MODEL files contain names for the otherwise unassigned registers (R0,R1,...,R15). The name for each register is its hexadecimal address (F0,F1,...,FF). If your program contains another name for one or more of these registers, only the Fx name will display in disassemblies. However, note that the variable names can still be accessed symbolically (e.g., in the "Display/Alter | Var/Reg" pull-down).

If you would rather see the variable names from your program in disassemblies, you can comment out those lines in the $MODEL file defining the Rn (Fx) registers. Do this by placing a comment character (pound-sign, "#") at the beginning of the A3 lines in the $MODEL file for those Rn (Fx) registers to be "eliminated".

Alternatively, if you would also like to see such variable names from your program displayed in the Register Window as well as in disassemblies, don't comment out the A3 lines in the $MODEL file. Rather, change the register name (the quoted string) in the appropriate A3 directive(s) from "Fn" to "*my_var*", where *my_var* is the variable name from your program.

Emulation Notes

The following characteristics apply to all *iceMASTER-COP8* emulators:

1) If a Break-point is set on an instruction, emulation will not break until <u>after</u> that instruction executes. Note that Break-points are independent of Pass-count-points.
2) If a Trace-On/Off-point is set on an instruction, tracing will be turned on/off immediately <u>before</u> the first cycle of that instruction.
3) If a Pass-count-point is set on an instruction, the pass counter is decremented and tested <u>after</u> that instruction executes. When the pass counter has decremented to zero (0) a break will occur when the next Break-point is reached.
4) <u>880 Probe Cards</u>. Upon reaching a Break-point, the timer is shut off approximately 20 cycles after emulation stops. When emulation resumes, the timer is restarted approximately 16 cycles before emulation in the target application program actually begins.

   <u>884*xx*/888*xx* Probe Cards</u>. There is no delay in stopping or restarting running timers upon reaching a Break-point or when emulation resumes.
5) If you are using HALT mode, in order to avoid possible synchronization problems, always place two NOP instructions following the HALT mode instruction.

MKSHF Utility

The MKSHF.EXE ("MaKe Symbolic Hex File") utility program allows you to take maximum advantage of *iceMASTER*'s symbolic debugging capabilities. MKSHF merges the symbol file (.SYM) generated by the COP800 Assembler with the code (.HEX) file generated by National's LMHEX utility. The resultant file is in a format suitable for loading into the *iceMASTER-COP8* system during a debugging session.

MKSHF ignores the typing information present in the .SYM file. Only the symbol's name and value (address) are extracted. By default, unless told otherwise, MKSHF assumes that a symbol is a code label (i.e., relative to the code memory space). You can optionally provide MKSHF an input file containing a list of "exceptions". It is in this way that you can denote:

1) the variable names (relative to the internal data memory space),
2) the numbers ("equates" – relative to no memory space in particular), or
3) those symbols which are to be totally ignored.

A sample batch file for producing a symbolic hex file, loadable into *iceMASTER*, follows:

```
rem File:   DEMO_C8.BAT
rem 1.  Assemble to generate ".LM", ".LST"  and ".SYM":
ASM800 demo_c8.asm /O=demo_c8.lm /L=demo_c8.lst /S
rem 2.  Convert ".LM" (load module) to ".HEX" (standard HEX):
LMHEX          demo_c8.lm
rem 3.  Create the Symbolic Hex File (".SHF"):
rem       Input:   demo_c8.sym (created by 1. above)
rem       Input:   demo_c8.exc (Exception File – created previously)
rem       Input:   demo_c8.hex (created by 2. above)
rem       Output:  demo_c8.shf
MKSHF  demo_c8.sym demo_c8.exc demo_c8.hex demo_c8.shf
rem 4.  Then, when you are using iceMASTER (ICE.EXE), select the
rem     "File,Load" pull-down command to load the file "demo_c8.shf"
rem     into the emulator.
```

The syntax for invoking MKSHF is:

MKSHF <sym_in> <excep_in> <hex_in> <hex_out>

where

<sym_in>

(input file) The ".SYM" symbol file generated by the COP800 Assembler. You must specify the "/S" option to the Assembler to create this file and you must be using "REV:E,22 JUN 90" (or later) of the Assembler.

<excep_in>

(input file) This file contains the list of "exception symbols". If a particular symbol does not appear in this Exception File (and it does appear in the <sym_in> ".SYM" file), that symbol will be treated as a code label. Each line in an Exception File should contain a symbol name followed by a single-digit code telling MKSHF how to treat the symbol:

File Entry	Meaning
<sym> 2	variable name (data memory label)
<sym> 5	number (not associated with any memory space in particular)
<sym> 9	completely ignore this symbol (do not output it to the ".SHF" file)

Once you create an Exception File for a particular application program, you needn't worry about changing that Exception File unless you add new data locations (variables) to your assembly language program and would like to access those variables symbolically in *iceMASTER*.

If you do not want to supply an Exception File, you must specify the argument as a single dash (–). If you do this, all symbols in the <sym_in> (".SYM") file will be treated as code labels.

To make it easier to create an Exception File, you can start with a copy of the supplied Exception File, "MKSHF.EXC". This file contains specifications for all symbols normally found in the standard COP800 Assembler include files (COP*xxx*.INC). Then, add the appropriate symbols in your assembly language program to that copy of MKSHF.EXC. If a symbol appears more than once in an Exception File, the last specification will be used. Thus, you should probably always add new symbols at the end of an Exception File.

<hex_in>

(input file) The ".HEX" file generated by LMHEX.

<hex_out>

(output file) The generated Symbolic Hex File (".SHF"). Note that the MKSHF program does not supply the ".SHF" file extension – whatever you want to use is fine.

All COP8 probe cards have the same basic layout as shown in Figure 1:

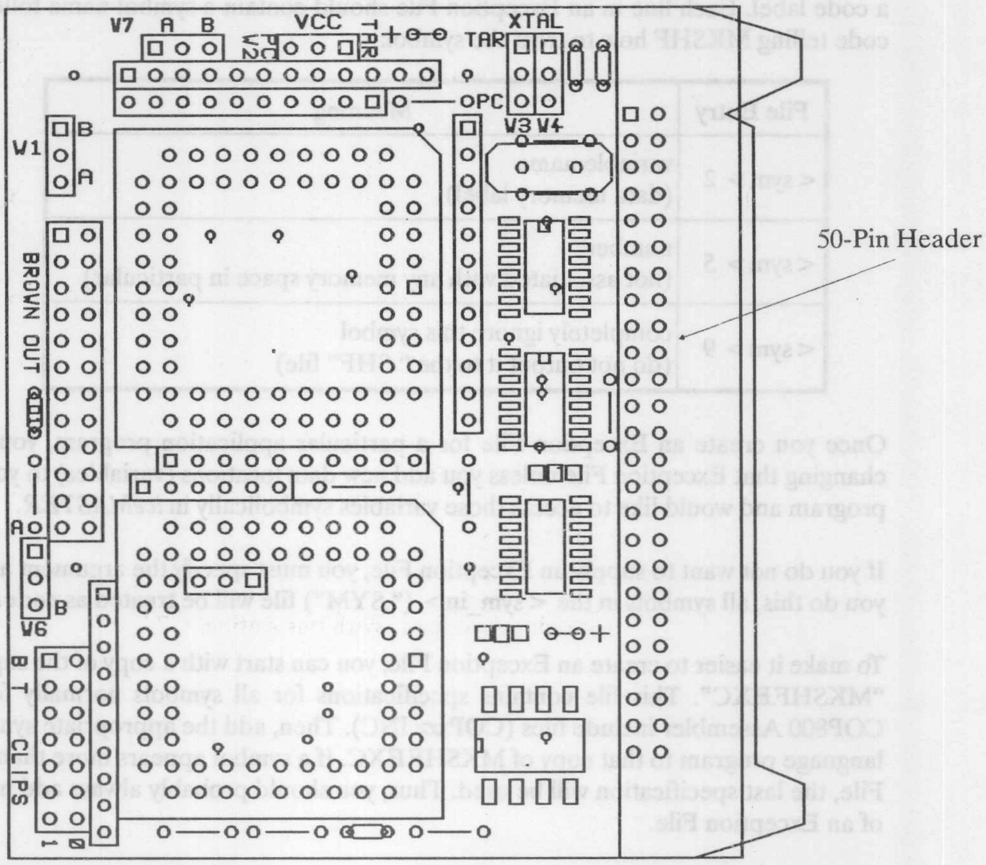


Figure 1. COP8 Probe Card

Jumper Blocks

The user-selectable jumper blocks on the probe card are:

W1 – SELECT

The function of the **SELECT Jumper** depends on the probe card. The configuration of this jumper is readable by the *iceMASTER* host software and its setting can be verified by selecting the "Configure | Identification" pull-down menu.

880. The **SELECT Jumper** determines how much on-chip internal RAM is available; this allows you to emulate the 820, 820CJ and 840 devices, as well as the 880. When the shorting block is placed between the center post and the "A" post, 64 bytes of RAM can be used. When the shorting block is placed between the center post and the "B" post, 128 bytes of RAM are available.

884*xx*/888*xx*. The **SELECT Jumper** allows the probe card to accurately emulate COP8 Mask Options. The W1 jumper controls the HALT Enable mask option. When the shorting block is placed between the center post and the "A" post, the HALT functionality of the microcontroller is disabled. When the shorting block is placed between the center post and the "B" post, the HALT functionality is enabled.

W2 – VCC

The **VCC Jumper** is effective only on the 2.3V-6.0V and 2.5V-6.0V probe cards. This jumper allows the operating voltage of the probe card to be supplied from either the probe card (PC) or the target system (TAR). The PC setting supplies only 5.0V and is required for out-of-target emulation. The TAR setting uses the voltage supplied to the VCC pin of the microcontroller socket on the target system. This voltage may range anywhere from 2.3V to 6.0V or 2.5VDC to 6.0VDC depending on your probe card.

On 4.5V-5.5V probe cards this jumper is hardwired to the PC setting. **THIS MUST NOT BE CHANGED! OPERATING A 4.5V-5.5V PROBE CARD OUTSIDE THIS VOLTAGE RANGE MAY PERMANENTLY DAMAGE THE PROBE CARD.**

When operating a 2.3V-6.0V or 2.5V-6.0V probe card with a voltage source less than 4V, the maximum operating frequency must be reduced per the National Semiconductor specification for the particular part. Attempting to operate the probe card above that maximum frequency will cause unpredictable results.

W3,W4 – XTAL

The **XTAL Jumpers** allow the clock source (CKI) to be supplied from either the probe card (PC) or the target system (TAR). An oscillator circuit (see **Miscellaneous Notes** on page 8) is used on the probe card to emulate the oscillator circuit of the microcontroller. When the PC setting is selected, a 10MHz crystal is connected to the oscillator circuit. With this setting, G7 is still available to the target system as a general-purpose input. When the TAR setting is selected, either a target crystal or clock driver can be used.

Oscillator Source	W3	W4
Probe Card	PC	PC
Target	TAR	TAR

D1 – BROWN OUT

The **D1 LED** indicates that the probe card processor is being run from a voltage source that is less than the Brown Out voltage for the probe card. The nominal Brown Out voltage is 2.34 VDC for the 820CJ probe card and 2.52 VDC for all other probe cards, with approximately 140 millivolts of hysteresis. This voltage level is due to the operating characteristics of the probe card and may differ from final production parts. Note that Brown Out is a feature of the probe card (not the device) and is therefore available on all probe cards. Currently, the only production part that incorporates Brown Out is the 820CJ.

W6 – Brown Out Disable

The **W6 Jumper** allows the user to emulate the Brown Out bondout option. When enabled, the Brown Out feature will be emulated. When disabled, the Brown Out feature will NOT be emulated. NOTE: the BROWN OUT indicator (the yellow LED on the probe card) will always light when a voltage lower than the Brown Out voltage is detected, regardless of the state of the W6 jumper.

Brown Out	W6
Disabled	A
Enabled	B

W7 – G7 Option

The **W7 Jumper** allows the user to emulate the G7 bondout option of the target processor. When emulating CKO, the probe card oscillator is driven out the target G7 pin. When emulating the INPUT/HALT bond out option the target G7 pin is routed directly to the Probe card processors G7 pin. When running with a target crystal W7 must be in the CKO (B) position.

G7 Option	W7
CKO	B
INPUT/HALT	A

Miscellaneous Notes

Probe Card Oscillator Circuit. Figure 2 is a schematic diagram of the oscillator circuit used on the COP8 probe card. The circuit uses an inverter as a Pierce oscillator amplifier to simulate the amplifier of the microcontroller.

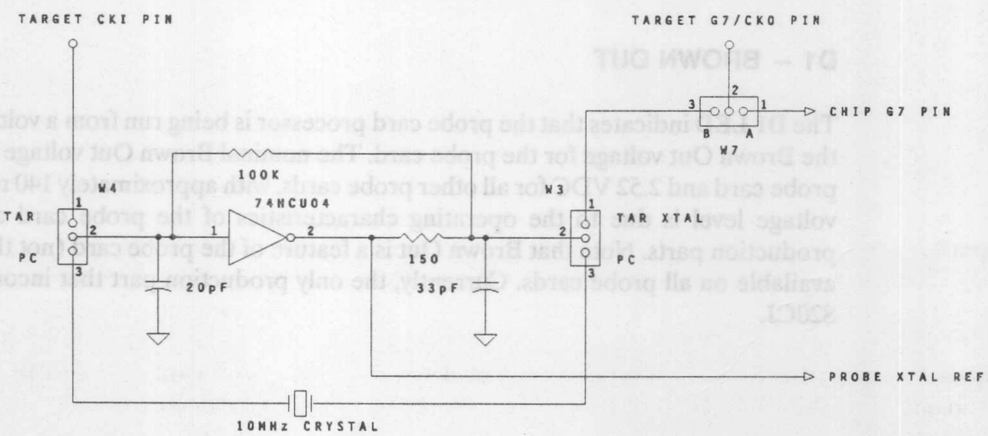


Figure 2. COP8 Probe Card Oscillator

When W3 and W4 are jumpered to the "PC" setting, a 10MHz crystal is connected to the amplifier. The output of this circuit is buffered before driving the probe card's microcontroller CKI input. This allows the CKO/G7 pin to be available to the target system as the G7 input.

When W3 and W4 are jumpered to the "TAR" setting, the target system may provide either a crystal or clock driver input. To verify that a target system crystal is operating correctly with the probe card amplifier, view the signal on pin 11 of the 68-lead PLCC device nearest the W3, W4 jumper blocks. Pin 11 is the clock input to the microcontroller. The signal should be at the target crystal frequency with standard CMOS voltage levels. If the signal does not exhibit these characteristics, the resistor and capacitor values on the target system crystal circuit should be adjusted accordingly.

<u>888CF VREF and AGND Pins</u>. The 888CF VREF and AGND signals are routed directly to the small adapter board pins (on the underside of the probe card) and are not connected to a source or ground on the probe card. As a result, when operating the probe card out of a target system (or in a target system without VREF and AGND), the I/ACH functionality of the 888CF microcontroller will not operate properly.

<u>Low Frequency Operation</u>. At crystal speeds below 500KHz, the A49 directive in the $MODEL file may need to be increased from the default value of 6 to a new value around 20. This is dependent on the speed of the computer running the host software and the crystal speed in the target system. Common indications of a need to increase the timeout value will be inability to establish communications with the *iceMASTER* base, or loss of communication after a break-point or host-break. The A49 directive is approximately 550 lines down from the top in the $MODEL file.